SYS-CON MEDIA

INSIDE: FREE WEB SERVICES SECURITY POSTER!

December 2003 Volume 3 Issue 12

WebServices JOURNAL

.NET J2EE XML

WSJ2.COM

International Conference & Expo

edge 2004 EAST

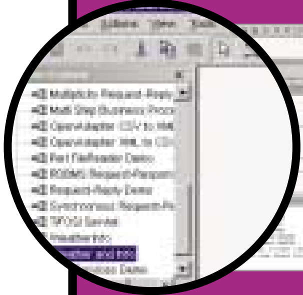
page 56

FEBRUARY 24-26, 2004

Hynes Convention Center, Boston, MA

Register By December 19, 2003

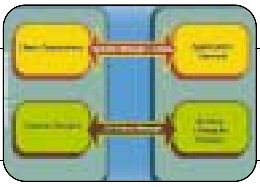
SAVE Up To $400



PAGE 8

DEVELOPING WEB SERVICES WITH OPEN SOURCE

FOCUS ON PORTALS/OPEN SOURCE

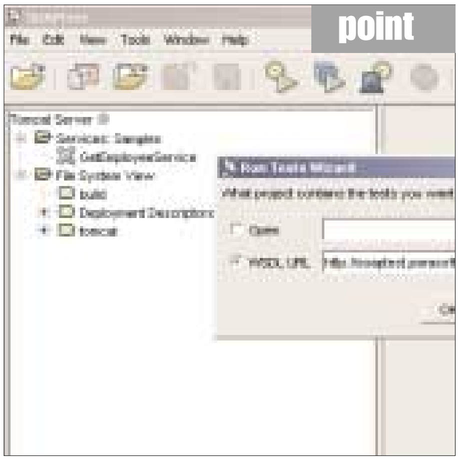


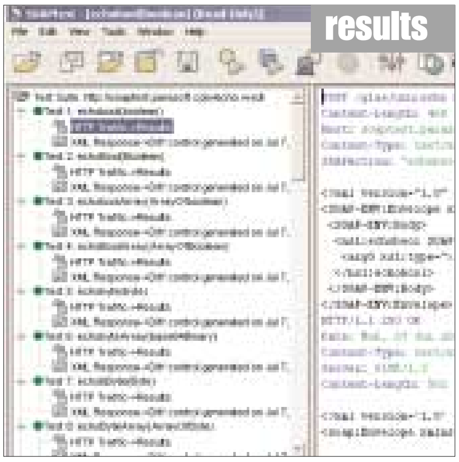
RETAILERS PLEASE DISPLAY UNTIL JANUARY 31, 2004

$6.99US $7.99CAN

PARASOFT
We make software work.™

Verify Web services instantly with Parasoft SOAPtest.





(It's as easy as 1-2-3.)

Parasoft SOAPtest is the first and only testing tool that enables you to instantly verify that your Web services are working properly.

Just point SOAPtest to your Web services WSDL files. It automatically creates test suites and runs comprehensive tests of the Web service's functionality and load capacities.

Server emulation for SOAP client development.
If you're developing a client-side application and need to verify functionalities such as credit card processing, oftentimes the Web services server is in use or otherwise unavailable for testing. Parasoft SOAPtest solves this problem by enabling you to emulate server responses and verify client functionality in just seconds.

SOAPtest also lets you quickly create templates written in Java™, JavaScript, and Python.

For Downloads go to www.parasoft.com/wsj12. Or call 888-305-0041.

Now supports WS-I Basic Profile 1.0

Compatibility:
Works with any SOAP-compliant implementation, including Microsoft and IBM
Windows 2000/XP
Linux
Solaris

A part of Parasoft Automated Error Prevention (AEP) Solutions and Services

PARASOFT®
soaptest®

WebServices JOURNAL .NET J2EE XML

INTERNATIONAL ADVISORY BOARD
Andrew Astor, David Chappell, Graham Glass, Tyson Hartman, Paul Lipton, Anne Thomas Manes, Norbert Mikula, Frank Moss, George Paolini, James Phillips, Simon Phipps

TECHNICAL ADVISORY BOARD
Bernhard Borges, JP Morgenthal, Andy Roberts, Michael A. Sick, Simeon Simeonov

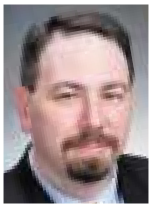
EDITORIAL
EDITOR-IN-CHIEF
Sean Rhody sean@sys-con.com
INDUSTRY EDITOR
Norbert Mikula norbert@sys-con.com
PRODUCT REVIEW EDITOR
Brian Barbash bbarbash@sys-con.com
.NET EDITOR
Dave Rader davidr@fusiontech.com
TECHNICAL EDITORS
Andrew Astor aastor@webmethods.com
David Chappell chappell@sonicsoftware.com
Anne Thomas Manes anne@manes.net
Mike Sick msick@sys-con.com
EXECUTIVE EDITOR
Gail Schultz gail@sys-con.com
EDITOR
Nancy Valentine nancy@sys-con.com
ASSOCIATE EDITORS
Jamie Matusow jamie@sys-con.com
Jean Cassidy jean@sys-con.com
ASSISTANT EDITOR
Jennifer Van Winckel jennifer@sys-con.com

PRODUCTION
PRODUCTION CONSULTANT
Jim Morgan jim@sys-con.com
LEAD DESIGNER
Richard Silverberg richards@sys-con.com
ART DIRECTOR
Alex Botero alex@sys-con.com
ASSOCIATE ART DIRECTOR
Louis F. Cuffari louis@sys-con.com
ASSISTANT ART DIRECTOR
Tami Beatty tami @sys-con.com

CONTRIBUTORS TO THIS ISSUE
Jordan Anastasiade, Ed Anuff, Brian Barbash, David Burdett, Alejandro Danylyszyn, Bill DeForeest, Alec Graziano, Rickland Hollar, Chris Peltz, Sean Rhody, Claire Rogers, Jothy Rosenberg, Scott Rosenbloom, Cesare Rotundo, Jim Webber

EDITORIAL OFFICES
SYS-CON MEDIA
135 CHESTNUT RIDGE ROAD, MONTVALE, NJ 07645
TELEPHONE: 201 802-3000 FAX: 201 782-9637
WEB SERVICES JOURNAL (ISSN# 1535-6906)
Is published monthly (12 times a year)
By SYS-CON Publications, Inc.
Periodicals postage pending
Montvale, NJ 07645 and additional mailing offices
POSTMASTER: Send address changes to:
WEB SERVICES JOURNAL, SYS-CON Publications, Inc.
135 Chestnut Ridge Road, Montvale, NJ 07645

A Face in the Crowd



WRITTEN BY
SEAN RHODY

Every now and then, I feel like two separate people. On one hand, I want to talk about services, pure and simple. I don't want to clutter it all up by discussing how to present the service to a user, or how to make it pretty, or how to make it cross platform. And yet, part of me realizes that there is a bigger picture to be considered.

While it doesn't get the press or the emphasis other parts of Web services do, the ability to finish the last mile, to put a human-accessible interface to the service, is an important part of what Web services are all about.

Most of the time, the user interface, or even the consumer of the Web service, is taken for granted as something about which we don't have to worry. And rightfully so, as Web services is first and foremost about plumbing.

Yes, I did say it's about plumbing. It's about connecting computer systems regardless of who made the system, what operating system it runs, and what programming languages are used. There are other things that go with it, as we've discussed over the course of this year within the magazine, such as business process management and security. But at its core Web services are about connectivity.

But somehow, some way, something has to use a Web service for something. And although in many cases the Web service is an intermediate part of a program, one that may never be directly interactive with a user, in some cases it will become necessary for the user to be able to invoke the service, even if indirectly.

Take a CICS transaction, for example (no, Sean, you take it). In many, many cases, a transaction represents the interaction of a 3270 screen, or its brethren, with a database. You could undoubtedly make a serious case for rewriting anything of that nature, but the reality for some organizations is that migrating hundreds, or thousands, of CICS transactions is not in the budget, and may never be. And while a 3270 screen is outmoded, the transactions behind it may not be. Putting a Web services interface on the transaction may enable multiple programs to get at it, but it still needs an interface.

That may not be the best example, but it certainly is a powerful one. The ability to put a modern face on legacy technology is definitely important. It's easy to discuss doing so in terms of deploying .NET, or a J2EE architecture with JSP pages. But the reality is that in some cases, the ability to directly expose a Web service using a simple mechanism would be extremely valuable.

Work is under way on some ways to do this, such as Web Services Remote Portal (WSRP). And portals are definitely a mechanism for providing services to a user, in convenient, bite-size packages. But since many companies don't have or use portal technologies, more than that is necessary.

It's debatable whether there needs to be a separate technology or standard for Web services user interfaces. It's difficult to imagine a broadly applicable technology, since the user interface is the last mile of the computing platform, and is intimately tied to the underlying desktop. Browser technology might be another answer, although it has its own problems with an anemic set of presentation tools. In most cases, we need to view the user interface as the domain of application programming languages and be thankful that most of the current generation of tools is able to interface fairly easily with Web services. But still, it's not too much to hope for to have a way to define the look and feel of an interface for a Web service, in XML of course, and leave it to the underlying virtual machine (or similar technique) to define the specific platform implementation. And it would make the last mile so much easier to walk.

I can't decide. Can you? Let me know. And happy holidays.

About the Author
Sean Rhody is the editor-in-chief of *Web Services Journal.* He is a respected industry expert and a consultant with a leading consulting services company.
■■■ Sean@sys-con.com

Mindreef News

Volume 1 December 2003 www.mindreef.com

Mindreef, WS-I, and Interoperability

Q: What is Web Services interoperability, and why is it important?

A: Web services exist because there is a need for language and platform independent communication between computers. Ensuring interoperability between Web service components is essential to achieving this vision. The platform-independent specifications that make up the core Web services protocols (SOAP, WSDL, UDDI) are powerful, but expressive to the point of ambiguity. The same service can be represented with WSDL in numerous ways, and toolkits can encode the same message with SOAP in entirely different ways. Without commonly-followed guidelines that limit protocol use to an unambiguous subset, Web services would not interoperate.

Q: Who is the WS-I, and what is the Basic Profile?

A: The Web Services Interoperability Organization (WS-I) is an open industry organization that promotes interoperability among Web services across platforms, applications, and programming languages. Acting as a "standards integrator," the WS-I developed Basic Profile 1.0, a set of guidelines to limit the scope of what is acceptable Web service usage. The Basic Profile 1.0 describes how these core Web service specifications should be used to develop highly interoperable Web services. To make the Basic Profile 1.0 immediately useful, the WS-I recently released the WS-I Testing Tools, which enable Web services developers to generate a Basic Profile compliance report against any number of Web services artifacts.

Q: How is a Basic Profile compliance report used?

A: Generating a Basic Profile compliance report is excellent for pass/fail analysis. If you are developing a Web service, you can deploy a passing compliance report alongside your Web service to publicize its high quality, as well as support your interoperability claim. For clients, performing pass/fail analysis via a Basic Profile compliance report can aid in your choice of toolkit, and ensure that you are invoking Web Services with quality requests.

Q: Where does Mindreef fit in?

A: Mindreef understands the need for interoperable Web services and the Basic Profile. And while the WS-I testing tools may notify you of the existence of a problem, the report can be long and cumbersome, and specific errors give only a broad indication of where the problem might be. In an effort to help you discover and resolve your interoperability problems, Mindreef presents SOAPscope 3.0.

Q: What is SOAPscope and how does it go beyond the WS-I Testing Tools?

A: SOAPscope 3.0 collects messages and WSDLs in a variety of ways, and provides an intuitive framework for performing Web services diagnostics and analysis. SOAPscope is capable of analyzing Web service artifacts against a myriad of specifications and guidelines, including the WS-I Basic Profile, SOAP, and WSDL, as well as Mindreef Best Practices. SOAPscope provides an interactive UI to help solve problems, not just report them. As Web service artifacts are analyzed, SOAPscope pinpoints and highlights the XML fragments causing the problem, and presents it within a larger context, including extensive help, to assist you in solving the problem.

Q: How do I resolve errors as I discover them?

A: SOAPscope 3.0 addresses the resolution of Web services interoperability errors by providing a unified Web service diagnostic system, and supporting a process of iterative improvements. As artifacts are collected and automatically stored, they can later be examined, and analyzed for potential problems. If a problem is found with a SOAP message, you can edit the artifact, resend it, diff the result against previous results, and re-analyze the transaction. WSDLs can be viewed, dynamically invoked without writing code, diffed for changes, and re-analyzed as they evolve.

Q: Isn't analysis something that a tester does at the end of the project?

A: No. Of course you'll want to generate a compliance report at the end of your development to support your interoperability claim, but running analysis throughout the development process can quickly uncover latent bugs and issues at a time closest to when they were introduced. Uncovering those bugs automatically with intermittent analysis saves developers and testers from spending hours hunting down something that can be detected within seconds. SOAPscope 3.0 provides you with life-cycle value, performing analysis at any stage, and even generating the WS-I Basic Profile 1.0 compliance report when you are ready to deploy.

Q: Now that I've done all this, can I guarantee interoperability?

A: You can never guarantee interoperability, but by integrating analysis into your development and testing process, you can be assured that you've done everything possible to be interoperable.

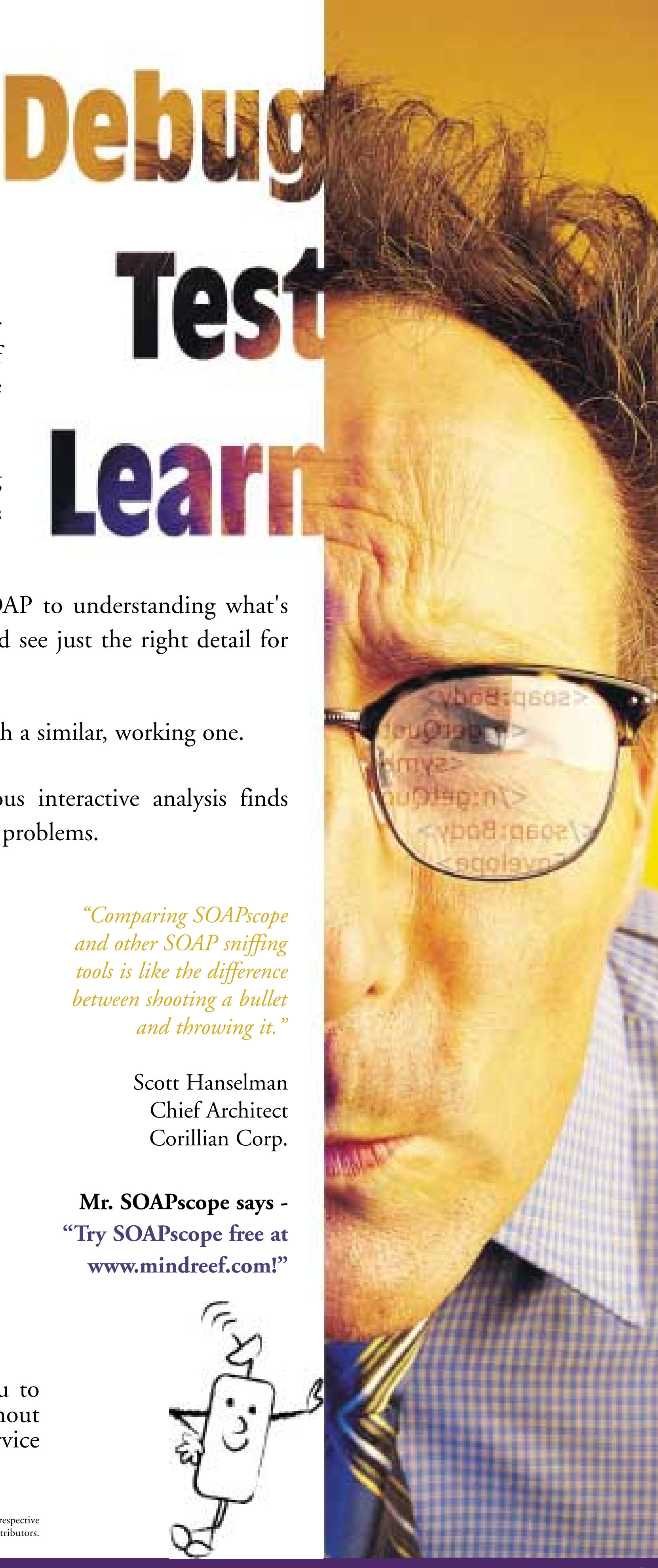
Q: Where can I learn more?

A: To learn more about integrating analysis into your development and testing process, and seeing specific, real world examples, stay tuned for Mindreef's forthcoming whitepaper on the subject. You can learn more about Mindreef by visiting http://www.mindreef.com, or by visiting the WS-I at http://www.ws-i.org.

"My purchase of SOAPscope has already more than paid for itself"
- Jim Albers

Introducing

SOAPscope 3.0



4 Ways to Know Web Services

Whether you are learning how a Web service works, or troubleshooting a tough problem, you need the help of a "smart" tool. SOAPscope leverages your intelligence and satisfies your curiosity by digging deeper, faster.

Try It Find patterns to solve problems by testing your Web service with different inputs without writing any code.

See It Look under the covers at WSDL and SOAP to understanding what's happening. Capture from any toolkit, and see just the right detail for the task at hand.

Diff It Compare a problem message or WSDL with a similar, working one.

Check It When the problem's not obvious, rigorous interactive analysis finds inconsistencies, errors, and interoperability problems.

Look What's New in 3.0

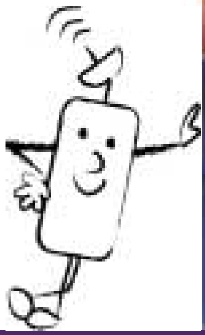
- Interactive Message Analysis
- Interoperability Testing System
- SSL Support
- HTTP Authentication Support
- HTTP Compression Support
- Visual Studio & .NET Integration
- Support for multi-byte encoding
- Best usability of any web services tool

The most comprehensive Web services diagnostic system available, and still **only $99!**

"Comparing SOAPscope and other SOAP sniffing tools is like the difference between shooting a bullet and throwing it."

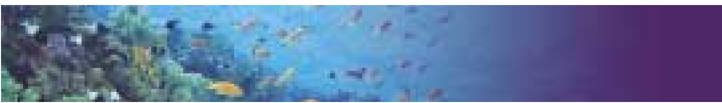
Scott Hanselman
Chief Architect
Corillian Corp.

Mr. SOAPscope says - "Try SOAPscope free at www.mindreef.com!"



Try It...at XMETHODS

Mindreef has partnered with XMethods to allow you to easily invoke hundreds of different web services without writing code. Just click on '*Try It*', next to any service name at xmethods.net.



"The Web Services Diagnostic Experts" Mindreef

Deliver dynamic PDFs from your J2EE application server

JClass® ServerReport

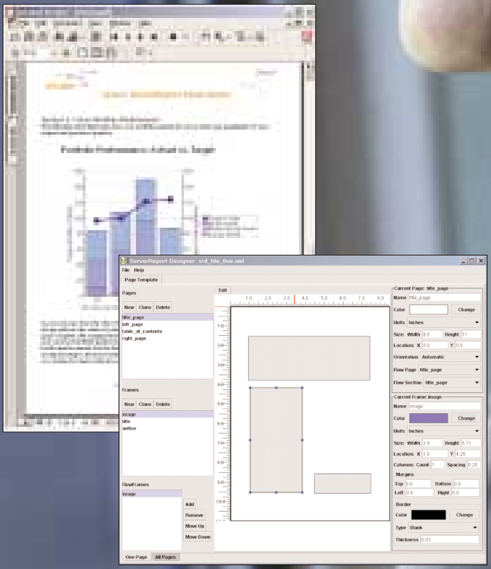
Bring high-quality printable documents, dynamically generated with up-to-the-minute data, to your Servlet, JSP and J2EE applications. Perfect for financial reports or e-billing statements, JClass ServerReport flows your content and data into customizable page templates, generating customized Adobe PDF documents that display and print anywhere.

100% Java class library for easy integration with your application server

Scalable design for high-volume web application environments

Now easier to use with new visual page design tool

New support for foreign-language content, including Asian characters and fonts



QUEST SOFTWARE®

Evaluate and experience JClass today - visit:

http://java.quest.com/jcsr/ws

WebServices .NET J2EE XML JOURNAL

PRESIDENT AND CEO
Fuat Kircaali fuat@sys-con.com
VP, BUSINES S DEVELOPMENT
Grisha Davida grisha@sys-con.com
GROUP PUBLISHER
Jeremy Geelan jeremy@sys-con.com
TECHNICAL DIRECTOR
Alan Williamson alan@sys-con.com

ADVERTISING
SENIOR VP, SALES & MARKETING
Carmen Gonzalez carmen@sys-con.com
VP, SALES & MARKETING
Miles Silverman miles@sys-con.com
ADVERTISING DIRECTOR
Robyn Forma robyn@sys-con.com
DIRECTOR, SALES & MARKETING
Megan Ring-Mussa megan@sys-con.com
ADVERTISING SALES MANAGER
Alisa Catalano alisa@sys-con.com
Carrie Gebert carrie@sys-con.com
ASSOCIATE SALES MANAGERS
Kristin Kuhnle kristin@sys-con.com

SYS-CON EVENTS
PRESIDENT, SYS-CON EVENTS
Grisha Davida grisha@sys-con.com
CONFERENCE MANAGER
Michael Lynch mike@sys-con.com
NATIONAL SALES MANAGER
Sean Raman raman@sys-con.com

CUSTOMER RELATIONS/JDJ STORE
CIRCULATION SERVICE COORDINATORS
Niki Panagopoulos niki@sys-con.com
Shelia Dickerson shelia@sys-con.com
Edna Earle Russell edna@sys-con.com
JDJ STORE MANAGER
Rachel McGouran rachel@sys-con.com

SYS-CON.COM
VP, INFORMATION SYSTEMS
Robert Diamond robert@sys-con.com
WEB DESIGNERS
Stephen Kilmurray stephen@sys-con.com
Christopher Croce chris@sys-con.com
ONLINE EDITOR
Lin Goetz lin@sys-con.com

ACCOUNTING
FINANCIAL ANALYST
Joan LaRose joan@sys-con.com
ACCOUNTS RECEIVABLE
Charlotte Lopez charlotte@sys-con.com
ACCOUNTS PAYABLE
Betty White betty@sys-con.com

SUBSCRIPTIONS
SUBSCRIBE@SYS-CON.COM
1-888-303-5282
For subscriptions and requests for bulk orders, please send your letters to Subscription Department
Cover Price: $6.99/issue
Domestic: $69.99/yr (12 issues)
Canada/Mexico: $89.99/yr
All other countries: $99.99/yr
(U.S. Banks or Money Orders)

WorldWide Newsstand Distribution
Curtis Circulation Company, New Milford, NJ

For list rental information:
Kevin Collopy: 845 731-2684, kevin.collopy@edithroman.com;
Frank Cipolla: 845 731-3832, frank.cipolla@epostdirect.com

SYS-CON Publications, Inc., reserves the right to revise, republish and authorize its readers to use the articles submitted for publication.

SYS-CON MEDIA

Ease Communication Between Portals and Back-End Systems

WRITTEN BY
ED ANUFF

IT managers are continually asked to do more with less. Competitive pressures and budgetary constraints compel IT departments to capitalize on the organization's existing infrastructure as well as maximize the value of any extensions or custom development efforts.

This has been especially true in the realm of Web portals, and the barriers to success are all the more challenging. Many organizations that run multiple, heterogeneous portal environments want to deliver pieces of application functionality as portlets that can be consumed through various internal and external portals. The custom development, reconfiguration, and integration work involved in deploying and redeploying portlets inside of multiple application servers can take tens of thousands of dollars worth of IT staff time.

All of which gives the IT industry ample cause to celebrate the long-awaited ratification of two complementary specifications – Web Services for Remote Portlets (WSRP) and JSR 168 – as industry standards. WSRP allows for "plug-and-play" of portals, intermediary content aggregation applications, and integration with applications from disparate sources. JSR 168, also known as "Portlets 1.0," is a Java Community Process portlet specification that defines a set of Java APIs to enable interoperability between portals and portlets.

As portal vendors far and wide adopt these new standards and begin to build support for them within their products, the potential for delivering low-cost composite applications to a single, contextual interface through the Web is more of a reality than ever. Building custom applications upon standards-based logic such as JSR 168 and WSRP will enable developers and IT managers to reuse application portlets in a variety of delivery platforms. This is a key element of a services-oriented architecture (SOA) in which software components can be exposed as services on the network and reused for different applications and purposes.

However, it's also important to recognize that the ability to more flexibly deploy portlet applications by writing standards-compliant linkages is just one piece of a successful portal initiative. Other crucial capabilities – such as how to utilize the portlet, permission it, customize it, and make it available to other portal sites – are handled by the portal management infrastructure itself. And that is where the rubber meets the road in terms of evaluating how capably the different portal software providers will leverage the power of WSRP and JSR 168 for their customers.

The challenge now for the vendors involved is to implement intuitive interfaces that allow organizations to take advantage of standards-compliant portlets within multiple portal sites in a cohesive and consistent way. Developers and IT professionals need to carefully scrutinize every vendor's performance and hold them accountable.

Some of the key criteria for IT organizations to apply in evaluating their current or potential portal software provider's capabilities for achieving interoperability between multiple portals in heterogeneous enterprise environments include:

- ***Dedication to supporting standards through delivered product features:*** Every vendor should be able to utilize Web services and JSR 168 portlets within the portal, including the ability to:
  - Discover SOAP-based XML data input and output parameters and to build custom application interfaces on top of those parameters.
  - Run portlets developed to the JSR 168 standard as well as portlets previously developed against a proprietary API as a means of providing backward-compatibility. Organizations will need a transition period in which the portlets may be written either to JSR 168 or the vendor's API.

- ***Ability to incorporate standards-based portlet functionality in a seamless and integrated way:*** How the vendor incorporates third-party or custom functionality inside the portal management system is as important as the ability to use portlets and portal applications developed in a standards-based way in the first place. Both the end user's and the administrator's experience of the portlets – regardless of their origin – should be completely transparent. Whether they are based on WSRP or JSR 168 standards or a vendor's native API, portlets should behave consistently and deliver key application functionality transparently to end users. Characteristics of this behavior and functionality include the following:

–Continued on page 50

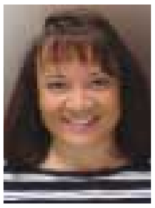
Developing Web Services with Open Source

A quick start for Web services technology

Over the past year, Web services have been positioned as a key enabler to application e-business integration. Many companies and vendors have made large investments in supporting the Web services development process. However, cost can pose a huge barrier for companies just beginning to investigate the value of Web services.

WRITTEN BY

CHRIS PELTZ &


CLAIRE ROGERS

Development shops can explore this emerging technology without making a large initial investment by developing Web services with open source platforms and development tools. Open source refers to the community of free applications and systems being written by developers around the world. With open source, the source code, its use, and redistribution cannot be forbidden by any organization.

The benefits of open source include the ability to view the source code, to see how features are implemented, and to modify the source to make changes. This allows developers to port tools to other operating systems and to build new products from open source code. A key theme with open source is flexibility, providing development organizations with the source code and the rights to modify it.

Web Services Development

Web services are built on a similar open premise. Web services standards such as SOAP and WSDL make software components available to any application developer. When combined with open source tools, the Web services development environment is widely available at a very low cost. This combination makes an attractive model for the evaluation of Web services as an enabling technology.

This article takes a look at both the Web services development process and the tools that can be used to get started quickly. Through the development of a prototype Web service, you can quickly understand how to evaluate these new technologies and their use within production scenarios. Our Web service provides weather forecast information for a specified Zip Code. While there are obviously many Web sites available for retrieving weather, we wanted to deliver this information as a Web service using SOAP.

The entire development was done on the Linux platform. In our prototype, the Eclipse environment with a Linux JDK was used to create the Java components. The Apache Axis platform was used to easily expose this Java component as a Web service. Apache Ant was then used for building and deploying the Web service, and PushToTest TestMaker for testing it.

Setting up the Environment

The first step in developing our service was the selection of a Linux distribution. While there are many Linux distributions available, we selected Debian Linux (www.debian.org). Debian is one of the most vendor-neutral Linux distributions, managed by the developer community. This distribution also has a strict open source–only policy, and does not contain any license-restricted code. Debian Linux has a successful model for creating high-quality, portable, and stable releases.

To deliver this service, we need a data source that can manage the weather forecast information. For this example, we decided to model the forecast information in a SQL database. The two most popular open source databases on the market today are MySQL and PostgreSQL. We selected MySQL (www.mysql.com) for this development primarily because of its ease-of-use and speed. After installing the product, we created the necessary tables for the Weather application. We designed two simple tables, one to main-

Ektron Success Story #1531

A leading baking products company needed to build a website that reflected brand and offered consumers access to up-to-date product information, recipes and promotional programs. With Ektron's Web content management solution they were able to achieve a high degree of site functionality and automation including the use of XML and Web Services to syndicate content to internal and third party web sites.

Enterprise Web Content Management without the enterprise integration.

ektron®

Let us show you how an Ektron XML Web Content Management Solution can enhance your Web site.

Learn More at:
www.ektron.com/ws

Ektron - Redefining Web Content Management

tain Zip Code information and another to hold the weather forecasts.

Another step required for the Java development was to select an appropriate Java Development Kit (JDK). The JDK was required to run a number of components in this example, including Eclipse, Apache Axis, and the application itself. The most popular Linux JDKs on the market today are the J2SE SDK from Sun, the Blackdown JDK (www.blackdown.org), and BEA WebLogic JRockit (www.bea.com). JRockit was selected for this example.

Developing a Java Application with Eclipse

Next we chose an integrated development environment (IDE). The IDE was used to develop the Java application for the Web service. Open source Java developers now have several choices when considering IDEs. Two of the more popular open source IDEs are NetBeans (www.netbeans.org) and Eclipse (www.eclipse.org). The Eclipse Platform was chosen because of its extensibility. This platform allows tool builders to independently develop tools that integrate with other developer tools. Figure 1 shows a screenshot from the Eclipse environment.

Eclipse operates with a set of views called *perspectives*. A perspective manifests itself in the selection and arrangement of editors and views displayed on the screen. For example, while developing a Java application, Eclipse changes to the Java perspective. This perspective sets up the GUI with the proper views and editors for developing Java applications. In addition, Eclipse includes a number of wizards to assist in the development of Java components. Figure 2 shows the wizard used for creating the base Weather class.

Next we added the code for the Weather service. The application takes a Zip Code and looks up the weather forecast for that area in the MySQL database using JDBC. Listing 1 is a partial listing of the code that retrieves this information (the code for this article is online at www.sys-con.com/webservices/sourcec.cfm).

Next we created the Forecast class. We created this class as a Value Object to hold the forecast information for the service. It provided a cleaner and more usable business interface for the caller. Rather than returning a simple String, we can return a more complex type describing the forecast information (see Listing 2).

We found that Eclipse brought together all of the tools we needed to successfully develop the service. Overall, Eclipse proved to be a very robust yet simple platform for building the Java components for the application. It has most of the same functionality as any other commercial tool, but it's also open source.

Installing and Configuring the Web Services Runtime

Since Web services are software components that are exposed to other applications via XML over HTTP, we needed a J2EE and Web services container in which to run our new service. For our runtime environment, we used Jakarta Tomcat (http://jakarta.apache.org/tomcat) and a Web services tool kit, Apache Axis (http://ws.apache.org/axis). Tomcat was downloaded from the Apache Web site. Once installed, we used a plug-in to Eclipse to start and stop the Tomcat server. We created our own Eclipse perspective for this so that we could reuse it for other applications running on Tomcat (see Figure 3).

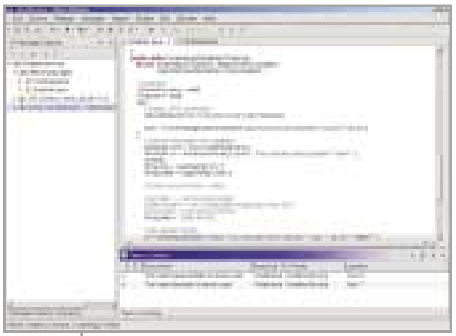


FIGURE 1 | **Eclipse**

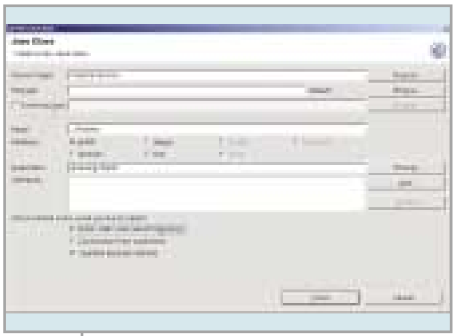


FIGURE 2 | **Adding a new Java class**

File Edit Source Refactor Navigate Search Project Tomcat Run Window Help
FIGURE 3 | **Eclipse with Tomcat plug-in**

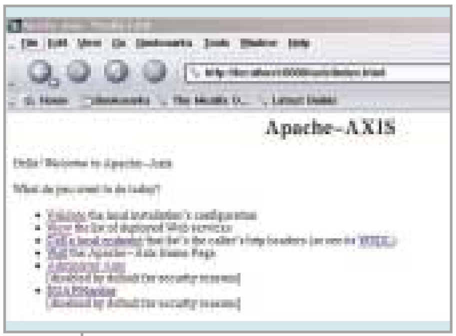


FIGURE 4 | **Apache Axis**

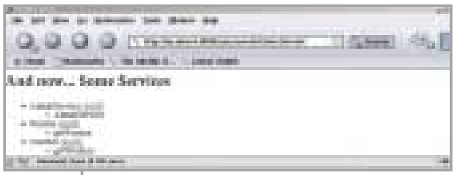


FIGURE 5 | **Checking service deployment**

In addition to the J2EE Web container, we needed a Web services environment. The Axis SOAP implementation includes a set of tools that help you build SOAP clients and servers. It provides support for both sending and receiving SOAP requests. We downloaded Apache Axis from www.apache.org/axis and used the latest release because of some added enhancements that support JAX-RPC and Java-to-WSDL mappings. Once installed, we tested Axis by viewing the URL http://localhost:8080/axis from Mozilla (see Figure 4).

Creating the Web Service Using Apache Axis

Our next major step was to create the Web service interface (WSDL) and related server-side bindings. A WSDL is an XML document that describes the data, messages, and operations that are exposed for a given service. The bindings allow us to easily map the WSDL interface to any back-end components.

Two approaches can be taken in creating a WSDL document. It can be created from scratch and later mapped to a set of back-end components. Or, a developer can start with their business logic and have the WSDL created automatically. We usually recommend starting from the WSDL for increased flexibility and probability. For this simple scenario, we used the automatic WSDL generation feature. Within Apache Axis, this can be accomplished through the Java2WSDL utility class:

```
java org.apache.axis.wsdl.Java2WSDL -o
Weather.wsdl –l "http://local-
host:8080/axis/services/weather" -n
urn:weather -p"Weather" urn:weather
weather.Weather
```

Listing 3 shows the WSDL generated running this command. It includes a set of data types, messages, operations, and SOAP bindings for the service.

The WSDL first describes the XML type used to represent the Forecast information. This is followed by the specific operations for the service, in our case getWeather. This operation is a basic request/response message, modeled with one input and one output message. The WSDL also contains the SOAP binding and service binding information. This binding specifies the URL location for accessing the service.

The next major step was to create the server-side bindings for the Web service. These bindings allow us to easily map the WSDL interface to the back-end Java components. Within Apache Axis, this is accomplished through an implementation class and a deployment descriptor. The following illustrates the invocation of the WSDL2Java class to generate these components:

```
$ java org.apache.axis.wsdl.WSDL2Java -
o . -s -p weather.ws Weather.wsdl
```

The WeatherSoapBindingImpl.java file generated contains an empty implementation for the service. We had to modify this code with the appropriate invocations to our original Weather class (see Listing 4).

We also had to create a new version of the Forecast class that implemented java.io.Serializable so that the data could be serialized as an XML stream. After that, we just needed to compile our code and generate a JAR file containing all of the class files. We copied the JAR into the Apache Axis lib directory so that Axis could locate the code.

The final step in the service creation process was to register the Web service with Apache Axis. Axis provides a deployment tool (AdminClient) to do this. The deployment descriptor specifies the service being deployed, the operations being exposed, and the mapping to the back-end implementation class. Listing 5 is a partial listing of the deploy.wsdd generated.

At this point, we successfully created, packaged, and deployed the service. We verified service deployment by viewing the list of services in a browser (see Figure 5).

Implementing a Build Process with Ant

The last section walked through the various steps required to compile, package, and deploy the Web service. This can be a very time-consuming process, especially if you must continually rebuild your Web service during testing. This is where a robust and automated build process can be very helpful in enhancing developer productivity.

One of the more popular open source build tools available today is Apache Ant. You can think of Ant as a next-generation Make utility, with the key difference that Ant is based on Java and XML. With Ant, you can compile and execute Java applications, create JAR files, and deploy files to Web directories. Apache Ant can also be used to assist in the generation of many of the Web services components.

While Ant can be downloaded from the Apache Web site, we decided to take advantage of the built-in integration between Eclipse and Ant. This integration allows you to take an Ant build file and run selected build targets. Supplied Eclipse plug-ins also provide a more robust editing environment for creating Ant build files (see Figure 6).

We focus here on how Ant was used to create the Java server-side bindings from the WSDL, and how that code is packaged and deployed to Apache Axis. Listing 6 provides a partial listing of the ANT file that was developed for this purpose.

In an Ant build file, a number of targets can be executed. A target represents a specific step in the build process. Our build includes an Ant target for creating the Java bindings from the WSDL. The build file also includes a target to compile the code using the <javac> tag. In the listing, we specify this tag, followed by the directory to compile and the CLASSPATH to use.

Once the code is compiled, we can package and deploy the service. In the ANT build file, we use the <jar> tag to create the Java archive. In this section, we specify the name of the JAR file to create, and the list of class files to package. We used the <copy> tag to copy the library to the Apache Axis lib directory. Finally, the AdminClient utility is invoked to register the deployment descriptor.

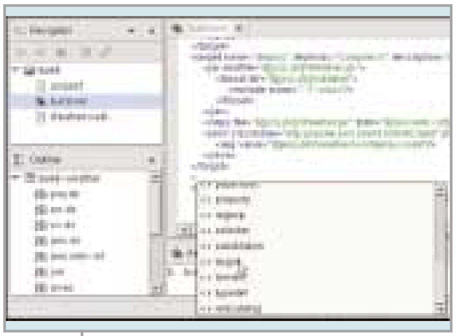
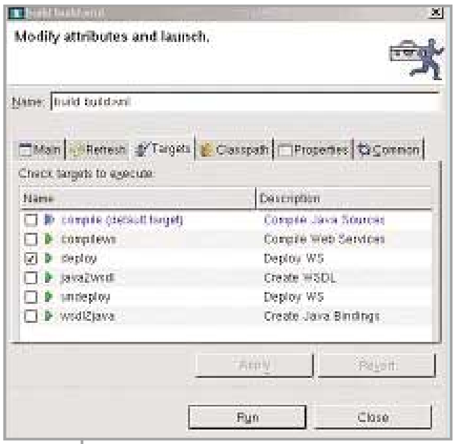


FIGURE 6 Building Ant files with Eclipse


Modify attributes and launch.

Name: build build.xml

Check targets to execute:

Name	Description
☐ compile (default target)	Compile Java Sources
☐ compilews	Compile Web Services
☑ deploy	Deploy WS
☐ java2wsdl	Create WSDL
☐ undeploy	Deploy WS
☐ wsdl2java	Create Java Bindings

FIGURE 7 Selecting Ant targets

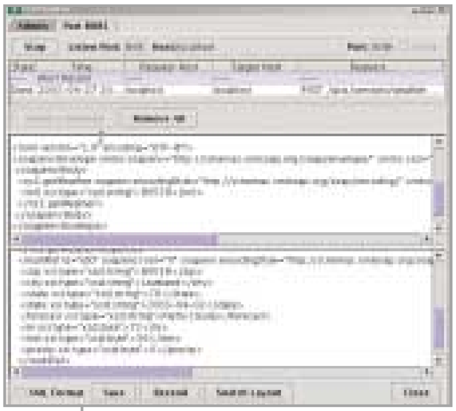


FIGURE 8 TCPMonitor GUI

After developing the complete build.xml file, we can execute specific build targets. From within the Eclipse environment, you can select specific targets to run (see Figure 7).

Overall, the use of ANT, combined with the integration into the Eclipse environment, provided a clean and elegant way to automate the build process for Java applications. It also provided an efficient mechanism to quickly build (and rebuild) the various Web services components.

“Open source refers to the community of free applications and systems being written by developers around the world”

Testing the Service

After deploying the Web service, the next step was to contact the Web service via a client proxy. A *client proxy* is a piece of code that communicates directly with the Web service, encapsulating the SOAP processing logic, and shielding the developer from having to write this code directly. Axis provides a mechanism for automatically creating the client proxy code through the WSDL2Java utility. We used the helper class, WeatherServiceLocator, that Axis automatically generated to locate the Weather Web service. Next, we inserted the code to bind to the getweather method in the WeatherServiceLocator. We communicated with the Web service to get a Forecast object with the appropriate information (see Listing 7).

To debug our Web service, we can use either the Axis tcpmon or SOAP Monitor utilities. tcpmon listens for connections on a given port on the localhost and forwards incoming messages to another port on another server. By inserting itself between the two ports, tcpmon can show you all incoming and outgoing SOAP messages. To use the tool, it was necessary to change our code to bind to port 8081 instead of 8080. Figure 8 shows how tcpmon can monitor SOAP traffic.

Axis also comes with a SOAP Monitor utility that can be used to monitor SOAP traffic without changing the port configuration. This utility is loaded as a Java applet within a browser window (http://localhost:8080/axis/SOAPMonitor). Overall, developers will find these utilities helpful in debugging a Web service, especially in cases where you get an exception or SOAP fault. You can determine whether the client properly constructed the SOAP message for the service.

Web services testing is another important development concern. While there are a number of open source testing tools available, such as JUnit and Anteater, we selected PushToTest TestMaker (www.pushtotest.com). This tool allows you to test the functionality, scalability, and performance of a Web service. The primary advantage of using TestMaker is its user-friendly development environment for creating test scripts. It comes with an object-oriented scripting language, Jython, as well as a library for communicating with Web services.

“the open source model helps these groups by allowing them access to a low-cost solution for Web services development”

When we used this tool, we encountered some issues with getting TestMaker to understand the Forecast XML type returned by the service. We had to add some additional logic to serialize the XML schema type into the JavaBean representation (see Listing 8).

Once the service is fully deployed and tested, a developer will need to consider monitoring the service for availability. This is where management tools are helpful, if not essential. For example, you could use the HP OpenView SPI for Apache Axis, which provides a mechanism to track the availability of the SOAP server. The HP OpenView Transaction Analyzer (OVTA) product, used to diagnose performance bottlenecks, is being enhanced to support Web services built with Apache Axis.

In the End

While we encountered several technical hurdles with this new development environment, we found these tools worked well together and were a boost to our Web services development productivity. In fact, we were quite surprised by some of the integration we found in Eclipse with Tomcat, Apache Axis, and Ant. We found the ability to locate, obtain, install, and use open source development environments and tools to be very straightforward. Not only did these tools work “out of the box,” but the quality was sufficient for our development purposes.

Development organizations need to quickly start using Web services technology, but can’t always afford to make significant early investments in tools that ultimately prove critical. However, the open source model helps these groups by allowing them access to a low-cost solution for Web services development. We were able to demonstrate a complete development process for Web services leveraging only open source tools. This should go a long way towards helping development teams get acquainted with Web services programming.

About the Authors

Chris Peltz and Claire Rogers are senior software consultants in HP’s Developer Resources Organization (http://devresource.hp.com).

■■■ chris.peltz@hp.com ■■■ claire.rogers@hp.com

A seasonal gift for your Web Services. With no strings attached.

Test the security of your Web Services with Vordel SOAPbox.
For free!

Vordel SOAPbox is a SOAP security test tool, and we'd like to give it to you as a seasonal gift. It's a free download. Not time-limited. Not a demo. Free. For you to use as often, and for as long, as you like.

- View SOAP requests and responses
- No coding or knowledge of XML security specifications required
- Send SOAP messages using existing security protocols (Client-side SSL, Basic HTTP Authentication)
- Generate XML Signatures
- Generate security tokens (SAML Assertions and WS-Security tokens)
- Send SOAP messages containing security tokens
- View file attachments in request and response
- Vordel SOAPbox automatically highlights security tokens in the message

Want to learn more about Web Services Security? Vordel SOAPbox also includes extensive help and documentation explaining the standards it supports, as well as its own functionality. Download Vordel SOAPbox from: **http://www.vordel.com/soapbox**

VORDEL® - the XML security company **www.vordel.com info@vordel.com**

A Roadmap for Web Services–driven BPM

Learning through example

In previous issues of ***Web Services Journal*** (Vol. 3, issues 7 and 10) we discussed how Web services-driven BPM presents an opportunity for new types of business solutions and explored the challenges to Web services business process management (BPM). This month, we provide a roadmap for success.

WRITTEN BY
ALEJANDRO DANYLYSZYN &

CESARE ROTUNDO

For optimal results, a roadmap should take into account not only the available technology, but also the prevailing industry standards and the internal characteristics of a company. Organizations must carefully weigh risk and reward, and align their processes to address each of the major Web services–BPM challenges. They must address technical challenges, such as lack of security controls at the protocol level and lack of transaction management capabilities, by leveraging available enterprise architecture technology and maturing WS standards. In our experience, service portfolio challenges, such as unstructured proliferation of services and lack of architectural layering, are effectively addressed through a cross-functional team referred to as a Center of Excellence (COE). The COE manages policy decisions, makes them operational and translates them into business solutions.

Following are two examples that illustrate how a good roadmap and COE can deliver concrete BPM–Web services solutions that successfully weave business processes.

Example One: Real-Time CRM

Web services-driven BPM enables long-spanning interactions that start and end with a customer. To deliver a real-time system for CRM and prevent it from becoming a functional silo, we recommend the following roadmap:

- Use the roadmap to create a process model that maintains transactional integrity by selecting products and standards to ensure that the system is transaction-driven.
- Roadmap your CRM architecture (i.e., the interfaces with existing systems) in the same way you merge multiple support organizations into one integrated team with standardized processes.
- Build business layers that break monolithic CRM applications into services clusters, externalizing components as dictated by the business process. Examples include:
  - An external Product Repository
  - An independent and scalable multi-channel order processor
- Build two common integration layers to "sandwich" the business functionality, isolating it from the core CRM application and from legacy systems and making it compatible with the other business processes in the enterprise.
- Apply the high-quality, incremental, process-based development model dictated by the COE to adhere to the business process evolution.
- Finally, since BPM facilitates the creation of a "true" real-time system by quickly and cost effectively dealing with business exceptions
  - Build a centralized subsystem to collect errors, where all systems can interface using Web services or more appropriate delivery mechanisms
  - Build automatic proceed/retry/abort mechanisms for controlled, predictable resolution capabilities while also providing error resolution tools to minimize the cost and maximize the reliability of human intervention

Example Two: Legacy Systems Application Transformation

Businesses today need to generate additional shareholder value while reducing investments in technology. This is particularly difficult for organizations experiencing competitive or regulatory pressures and that have their IT assets "locked-up" in costly, inflexible, and complex legacy applications. BPM and Web services enable companies to gain a competitive advantage by transforming millions of lines of outdated COBOL code into a flexible, service-oriented architecture (SOA). The recommended roadmap for companies in this position has three main steps:

- Avoid service portfolio pitfalls by reverse-engineering and documenting existing IT assets, including program hierarchy, job flow, data model, business rules, and context. Subject matter experts and application designers should then use this information to describe the capabilities of the target SOA in terms of discrete units of functionality (i.e., services.)
- Catalog the identified services by tier based on their purpose (for example client, presentation, business logic, integration, or resource.) If the target SOA addresses new functionality or constraints, identify and add the additional services to the catalog. Because the solution will operate within the boundaries of the enterprise architecture, technical challenges will be overcome by leveraging existing components.
- Define a value-based roadmap for deploying Web services within the SOA. For example, some services may remain within the boundaries of the legacy systems, wrapped with a Web service interface; others may be decommissioned legacy components rewritten in J2EE or .NET. In

> “Organizations must carefully weigh risk and reward and align their processes to address each of the major Web services–BPM challenges”

most cases, client- and presentation-tier services are migrated first, resource-tier services last.

These services become resources to BPM for creating, managing, and monitoring processes. Web services give BPM the potential to create nimble processes and applications by capturing events from a variety of sources and presenting them with a consistent interface.

Building BPM solutions that leverage Web services requires neither luck nor a stroke of genius, but rather is the result of methodical approaches. The opportunity is certainly there, and by creating a solid roadmap that identifies and overcomes the challenges, companies can reap the reward.

About the Authors

Alejandro Danylyszyn is a senior manager at Deloitte. He has worked for over 15 years as a consultant to large high-technology manufacturers, telecommunications carriers, and financial services companies in the areas of strategy, operations/process improvement, and solution design/implementation, with a focus on systems integration, enterprise portals, and Web services. Alejandro holds a master’s degree in software engineering from Carnegie Mellon University

adanylyszyn@dc.com

Cesare Rotondo is a senior manager at Deloitte. His expertise is in applying IT for business results and in managing large implementation projects around real-time business integration and customer integration solutions. His IT focus is around the enterprise software infrastructure, particularly EAI, B2Bi, enterprise portals, BPM, Web services, and J2EE. Cesare holds an MBA from INSEAD.

crotundo@dc.com

Will Standards Turn Portals into Commodities?

Two approaches that work

As a senior architect I always have a weather eye on evolving technologies in order to answer questions on how decisions made today will affect applications three to five years into the future. Occasionally, I hear or read a bellwether statement, one that makes me say, "I need to dig deeper into what was said in order to better understand its implications and track the underlying technology's evolution because it could have a significant impact on the way we are, or should be, developing applications." That was my reaction when I read Robert Brock's comments in the March 25, 2002, issue of *eWeek*: "If the [JSR 168] standard becomes successful, then the portal could become a commodity, just like Netscape or Internet Explorer."

WRITTEN BY
RICKLAND HOLLAR

When I started researching Brock's comment, I found not one but two standards proposals: JSR (Java Specification Request) 168, which is the Java Community Process's (JCP's) initiative for creating a standard application programming interface (API) between portals and portlets; and Web Services for Remote Portals (WSRP), OASIS initiative for standardizing the interface between portals and remote portlets. The standards are finally out in version 1.0 form, describing a set of interfaces that could in fact turn Brock's prediction into a reality. In this article, I'll look at the standards, the question of portals becoming commodities, and the importance that question's answer might have on your organization's decisions in writing applications today.

When Robert Brock made his comment about portals he was actually talking about "portal frameworks." What's the difference? A portal is a Web site; it is a collection of links, or windows, geared towards a particular set of content, workflow, transactions, or collection of systems. A portal's distinguishing characteristic is the type of content it provides:

- Mega portals aim to be one-stop gateways to the Web.
- Horizontal portals aggregate information across subject areas or industries.
- Vertical portals aggregate information within a subject area or industry.
- Enterprise portals aggregate information for an organization.

Portal frameworks provide the infrastructure and tools for building portal sites regardless of which type they might be.

A typical portal framework combines a number of different tools. First and foremost, it contains the tools needed to aggregate, organize, and present information through a Web browser. Portals from this perspective are analogous to icons on the Windows desktop at an operating systems level or database icons on Lotus Notes' desktop pages at an applications level: they provide a convenient way for users to organize information. In this role, portals provide a uniform look and feel while allowing users to customize and personalize presentation.

The user interface to a portal is a portal page, containing some number of portlets that users can arrange into columns and rows and minimize, maximize, or arrange to suit their individual needs. Each portlet is a window into a Web site or application. The portal framework can define a default common look and feel for the portlets appearing on the portal page, and is responsible for intercepting and routing URL requests into specific portlets and for supporting navigation both between portlets and, in some instances, within portlets.

In addition to managing aggregation and presentation, the portal framework provides the infrastructure for handling common services across portlets. Portal frameworks now include an increasing number of advanced capabilities, such as:

- Business intelligence
- Categorization
- Collaboration
- Content management
- Integration
- Knowledge management
- Search
- Security
- Reporting
- Wireless access
- Workflow

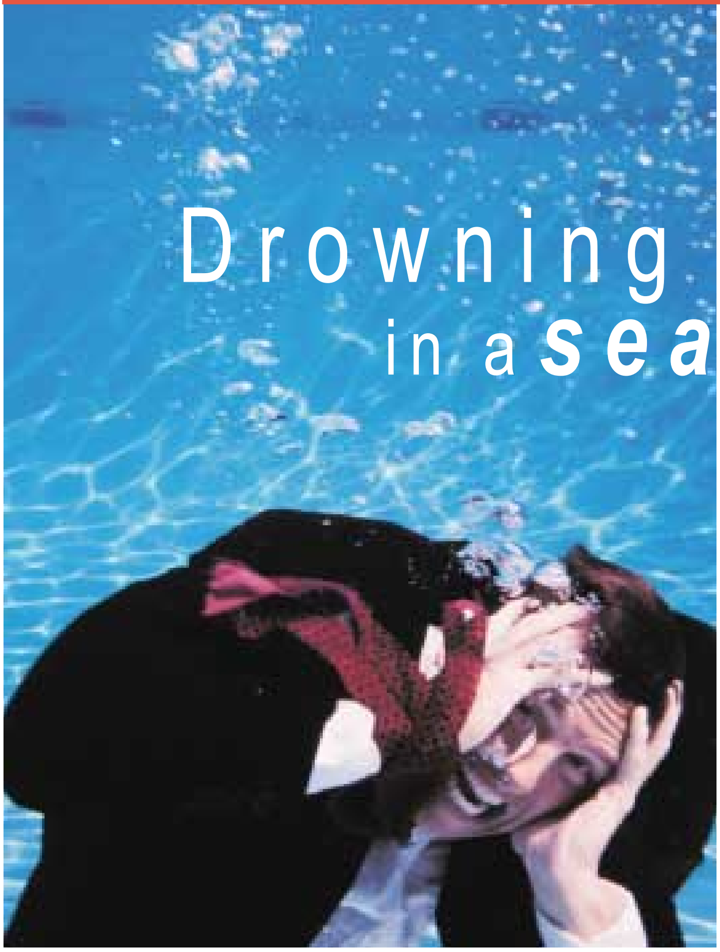
The framework may provide these capabilities natively or through third-party, add-in products. It is becoming increasingly difficult to determine where portal frameworks end and the advanced capabilities begin.

JSR 168

JSR 168 Version 1.0 became part of Java 2 in October with the Executive Committee for SE/EE's approval of the 1.0 specification. JSR 168 creates a standard API for portlets to "plug into" Java 2 Enterprise Edition (J2EE)–based portal frameworks. The goal is to have this API provide applications with portability across portal frameworks – allowing an application written to work in one framework to be able to execute in any standards compliant framework.

JSR 168 defines a portlet as a Web component that is managed by a portlet container. The portlet container runs portlets

Introducing the StrikeIron™ Web Services Analyzer



Drowning in a sea of WSDL descriptions?

The StrikeIron™ Web Services Analyzer to the Rescue!

Discover, invoke and understand any Web service in just a few clicks.

The StrikeIron Web Services Analyzer provides software developers, website builders and business analysts the ability to locate and understand the data structure, data requirements, behavior and results of any Web service anywhere in the world. All the details you need to make use of a Web service and exploit it per a given business scenario are available at your fingertips.

The StrikeIron Web Services Analyzer allows you to:
* Visualize any Web service in a few clicks
* Invoke any Web service and fully understand its behavior
* Cut development time from hours to minutes

The Web Services Analyzer is the first in a suite of StrikeIron products that will make Web services the first choice of development shops and business users who are implementing Service-Oriented Architectures or are taking advantage of Web services with traditional applications such as Web sites and business intelligence solutions.

STRIKE IRON

Delivering the World of Web Services℠

Download the Analyzer today at www.strikeiron.com and provides them with the necessary runtime environment. The portlet container may operate either as part of or independently of a portal application (framework), the portal application being responsible for common portal services. Figure 1 shows the JSR 168 Portal Reference Architecture, originally presented in Alejandro Abnelnur's Portlet Specification briefing to the OASIS Technical Committee (you can find the briefing in its entirety at the OASIS Web site, www.oasis-open.org), modified to illustrate these components. In this architecture, the portlet generates markup fragments that the portlet container provides to the portal framework; the portal framework adds a frame, a title, and control buttons to create a portal window that becomes part of a larger portal page.

The JSR 168 API provides standard interfaces that allow Java-based portlets to interact with the portlet container and portal framework. The API defines standard interaction points exposed through an extensible portlet interface. It includes classes and methods for:

- ***Life-cycle management:*** Enabling portlets to interact with the container during portlet construction, destruction, and initialization
- ***Presentation:*** Allowing portlets to detect and set window states and portlet modes, and render content through interactions with a preferences object within a portal window
- ***Personalization:*** Allowing portlets to set and access user preferences and profile information
- ***Security:*** Providing portlets access to user authentication and session information

While JSR 168 addresses the interfaces between local portlets and the portlet container, it leaves it to the vendor to work out the interfaces between the portlet container and the portal framework, and it looks to WSRP to address the issue of remote execution.

WSRP Version 1.0

OASIS approved WSRP Version 1.0 as an OASIS standard in August; they are now working on Version 1.1, which is due out next year. WSRP provides a component model that allows user-facing Web services to easily plug into portal frameworks as remote portlets. Figure 2 shows the WSRP Reference Portal Architecture (see the WSRP Overview briefing at the OASIS web site) that supports both local and remote portlets. Local portlets run on the portal server (framework) and interact with the portal framework through the Portal API (JSR 168 for J2EE-based frameworks). Remote portlets run on another server and interact with the portal server through Generic Portlet Proxies and WSRP. Generic Portlet Proxies allow the portal server to interact with remote portlets in the same way it interacts with its own local portlets, through the Portal API. WSRP provides the protocol for the portal server to interact with the remote server. WSRP also allows the portal server to make its local portlets available to other portals.

WSRP defines portal services as interactions between consumers and producers, where consumers are applications and portals that "consume" portlet services and producers are portal frameworks (or Web services applications) pro-

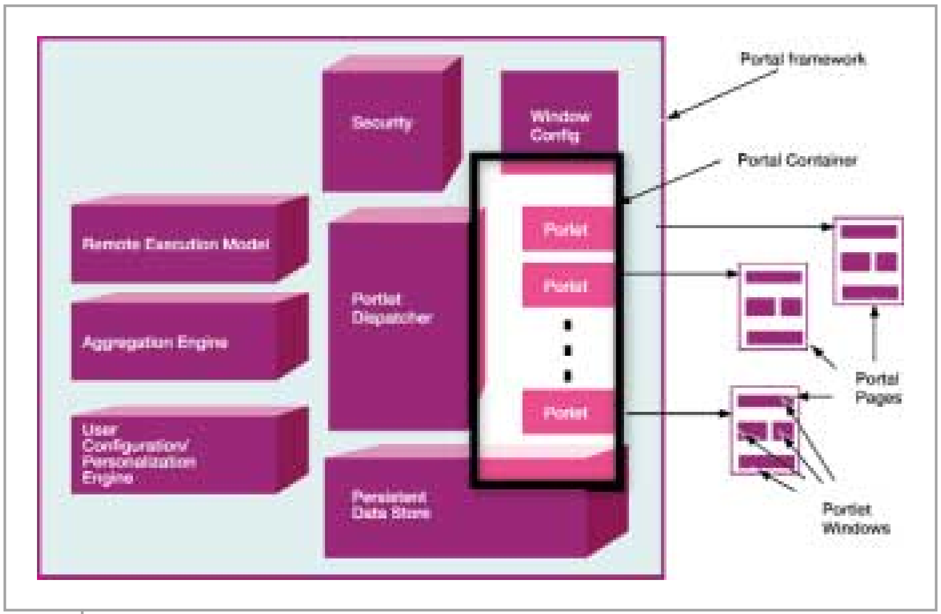


FIGURE 1 | JSR 168 Portal Reference Architecture

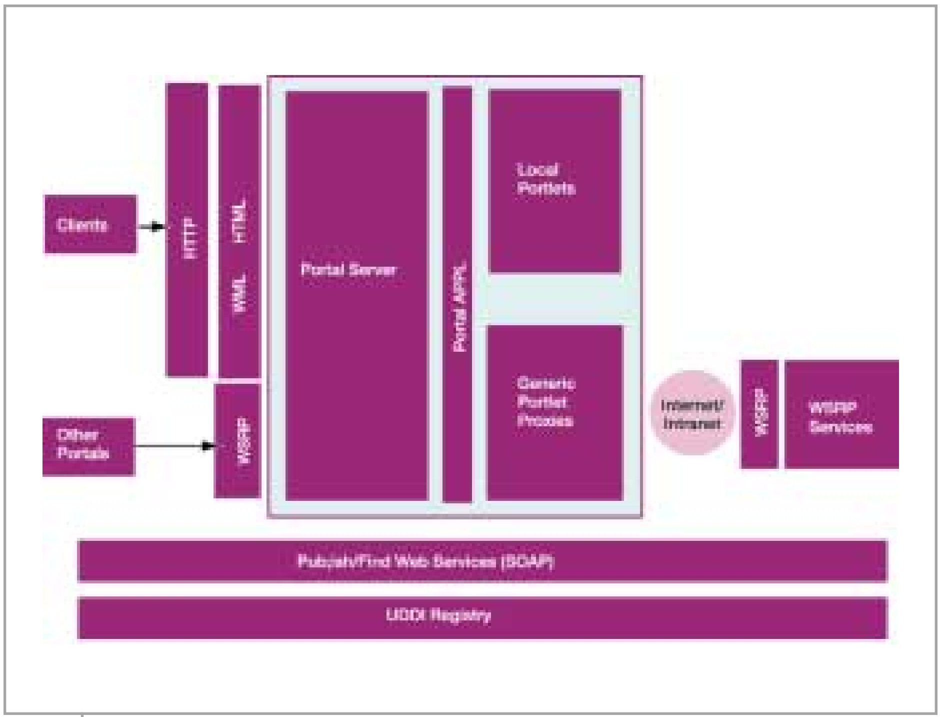


FIGURE 2 | WSRP Reference Portal Architecture viding those services. WSRP defines portlets as Web services that generate markup and permit consumers to interact with that markup, and WSRP producers as containers containing Web services portlets. In this model, portlet containers expose their framework and the local portlets it supports through four types of Web services interfaces:

- ***Service description:*** Enables consumers to learn about the producer's capabilities and its portlets
- ***Markup:*** Allows consumers to request and interact with markup fragments and to convey information about window modes and states
- ***Portlet management:*** Gives consumers access to portlet state and property information and influence over portlet life-cycle events
- ***Registration:*** Allows consumers to create relationships with producers; and to register, deregister, and modify relationship information

WSRP builds on Web services standards for publishing, finding, and binding services. Its goal is to enable any application, including portal frameworks, to publish their content as portlets, which portals can consume.

JSR 168 and WSRP are complementary: one's focus is on the local portlet code and its interactions with the framework while the other's is on remote interactions. Once both standards mature and appear in products, you will be able to develop portlet applications to the JSR 168 standard and depend on portal frameworks to provide WSRP services, when appropriate, as illustrated in Figure 3. This will allow you to create "write once, run anywhere" portlet applications without concern for whether users will access the portlets locally or remotely, through the framework or through another application.

JSR 168 and WSRP lay the foundations for future portal framework products. Brock's comment about portals becoming commodities implies a level of standardization that creates a certain degree of product agnosticism. This agnosticism could occur at two levels: at the market level determining the future of framework products themselves and at the individual developer level determining how you should be writing your particular applications. Are the standards enablers at either level? Yes. I've already discussed how they are enablers at the individual developer or application level.

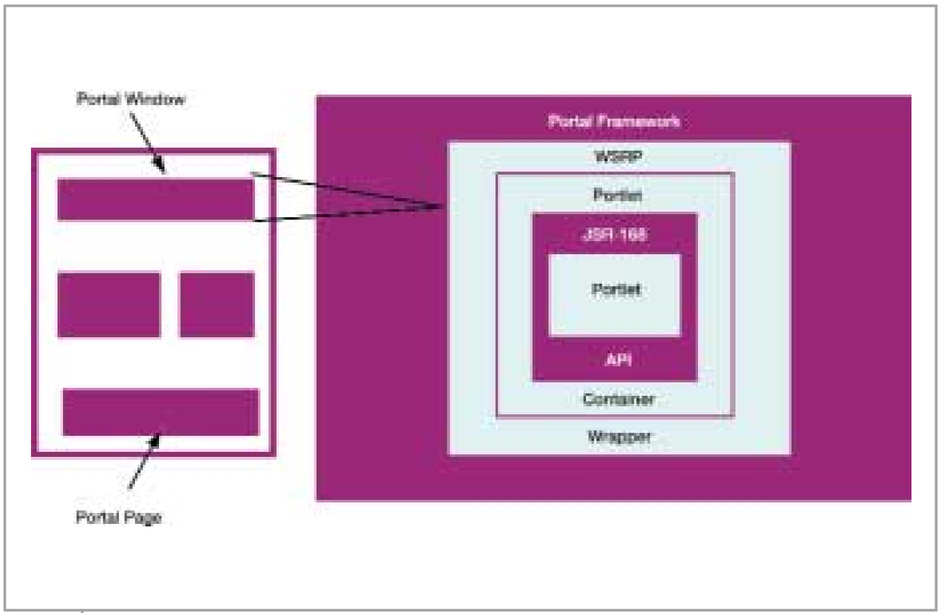


FIGURE 3 **Standards-based portlet**

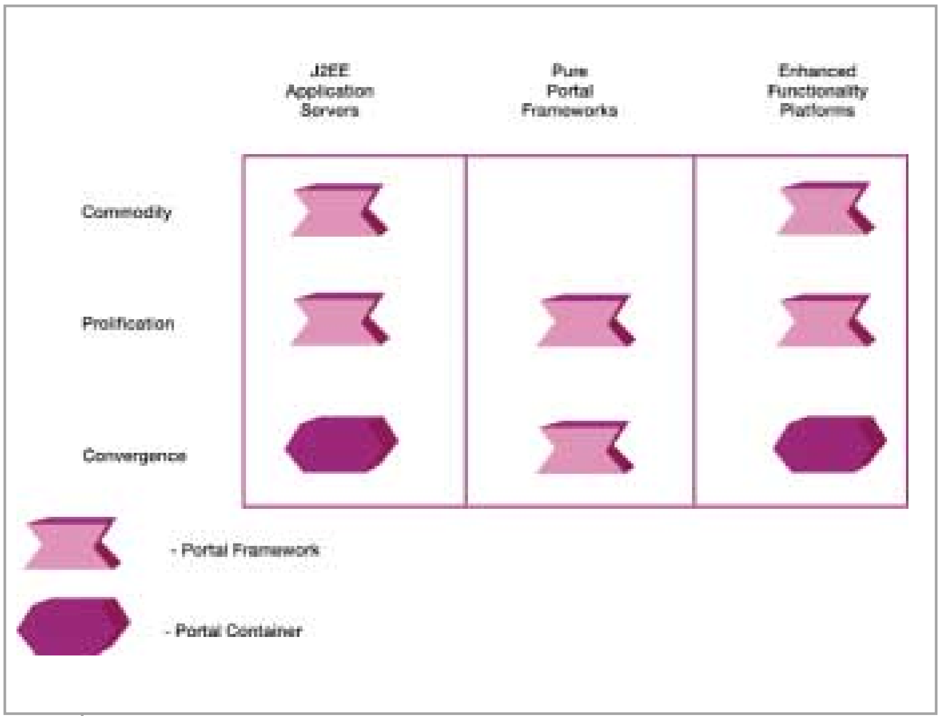


FIGURE 4 **Possible outcomes**

To answer the question at the market level, let's look at what it means to become a commodity at the market level and how the standards' scope might affect the outcome, and forecast possible outcomes and what they would mean to you in terms of how you develop applications.

First, what does it mean to say the portal framework could become a commodity? A commodity is a product or service offered at or below cost to drive the sales of other products. Products tend to become commodities when the cost is already low and price becomes the sole discriminator. When a product becomes a commodity, it tends to lose its identity as it is absorbed into other products. Most of us immediately think of the Web browser when someone talks about commodities and software. Netscape and

Microsoft "gave" browsers away in hopes of using them to sell other products. Could the JSR 168 and WSRP standards cause a similar phenomenon with portal frameworks, at least within the Java community? Could portal frameworks in this community become free or lose their identity?

Second, how does scope impact the answers to these questions? JSR 168 and WSRP focus on core portal functionality (aggregation and personalization) rather than advanced features (search, content management, knowledge management, and business intelligence), meaning they only address a subset of portal framework functionality. This opens the possibility for multiple classes (a high and a low end) of portal frameworks.

Finally, both standards distinguish between portlet containers and portal frameworks (leaving aggregation to the framework) and focus on defining the interfaces between portlets and portlet containers. This distinction opens the door for vendors to incorporate portlet containers, but not portal frameworks, into their products (similar to Web servers today where some Web servers are also J2EE Web containers).

Figure 4 summarizes the possible outcomes:

- ***Outcome 1: Portal Framework Commodities:*** Portal frameworks become commodities and lose their identities – application server products absorb them from below and Enhanced Functionality (content management, knowledge management, and business intelligence) platforms absorb them from above.
- ***Outcome 2: Portal Framework Proliferation:*** Portal frameworks retain their identity as separate products, but they continue to appear in both application servers and enhanced functionality products.
- ***Outcome 3: Portal Framework Convergence:*** Portal frameworks retain their identity as separate products, but disappear from application servers and enhanced functionality platforms because they switch from framework to container strategies.

Let's look at the likelihood of each.

Outcome 3, which is a return to pure-play portal products, is the least likely of the three to occur. Many application server and enhanced functionality vendors already include portal frameworks as part of their products today. Oracle and BEA Systems are examples in the application server market; BroadVision, PeopleSoft, and SAP are examples in the enhanced functionality platform market. JSR 168 and WSRP containers offer an alternative by making it easier to tie products together, increasing portability and interoperability, and potentially reducing product development costs, but the question becomes why would vendors who are successfully integrating frameworks today switch strategies, limiting themselves to providing only portlet containers. The answer is they wouldn't. Such a strategy would create dependencies in areas where they do not have dependencies today, while only marginally reducing costs. This would reduce their ability to discriminate their product from the competition's and prove too limiting.

Outcome 2 is the second least likely to occur. With both application server and enhanced functionality vendors integrating portal frameworks as part of their product offerings, the market for pure-play portal frameworks becomes very narrow and primarily price driven. The question becomes whether pure-play portal frameworks can survive in such a narrow space. For this scenario to play out, pure-play vendors would have to find ways, other than price, to distinguish their products; the need for product differentiation would drive these vendors to compete on enhanced functionality and features. You have to ask, at what point would increasingly function-rich, feature-laden products cease being portal frameworks and become either application servers or enhanced functionality platforms?

That leads us to Outcome 1 being the most likely of the three to occur. Application server vendors will incorporate JSR 168– and WSRP–compliant container capabilities into their products in order to maintain standards compliance. They will continue to include frameworks in order to differentiate their products and to promote their products' use in development environments. Others will follow in order to remain competitive. At the other end of the spectrum, enhanced functionality platforms will continue incorporating portal frameworks into their products in order to promote the enhanced functionality they offer and to provide full, independent solutions. They will include JSR 168 and WSRP containers in order to increase portability and interoperability. These trends' combined force will make price the primary discriminator for pure-play portal frameworks, making it unlikely they will be able to maintain their identity.

Conclusion

My weather eye sees the same future as Brock's: portal frameworks will in all likelihood become commodities (though it's debatable whether normal market pressures, rather than standards, aren't the real driving force). Given this prediction, you are probably asking what you should do to position your applications. First you should ensure you are working with vendors that have been part of the JSR 168 and WSRP standards efforts with strong commitments to integrating the standards into their products. Once the standards appear in the products you use, you should focus on writing portlets using the standards' defined APIs and interfaces, avoiding to the maximum extent possible any proprietary extensions. This should allow you to develop "write once, run anywhere" portlets, provided you stick with standards-based products. What if the prediction is wrong? How would this strategy change? It wouldn't! That's the beauty of standards and one of the benefits of the JSR 168 and WSRP efforts.

Regardless of which scenario actually unfolds, you should be able to develop "write once, run anywhere" portlets that make your applications more or less product agnostic and able to integrate with any of the standards-based frameworks. This promise is what following standards-based strategies is all about.

References

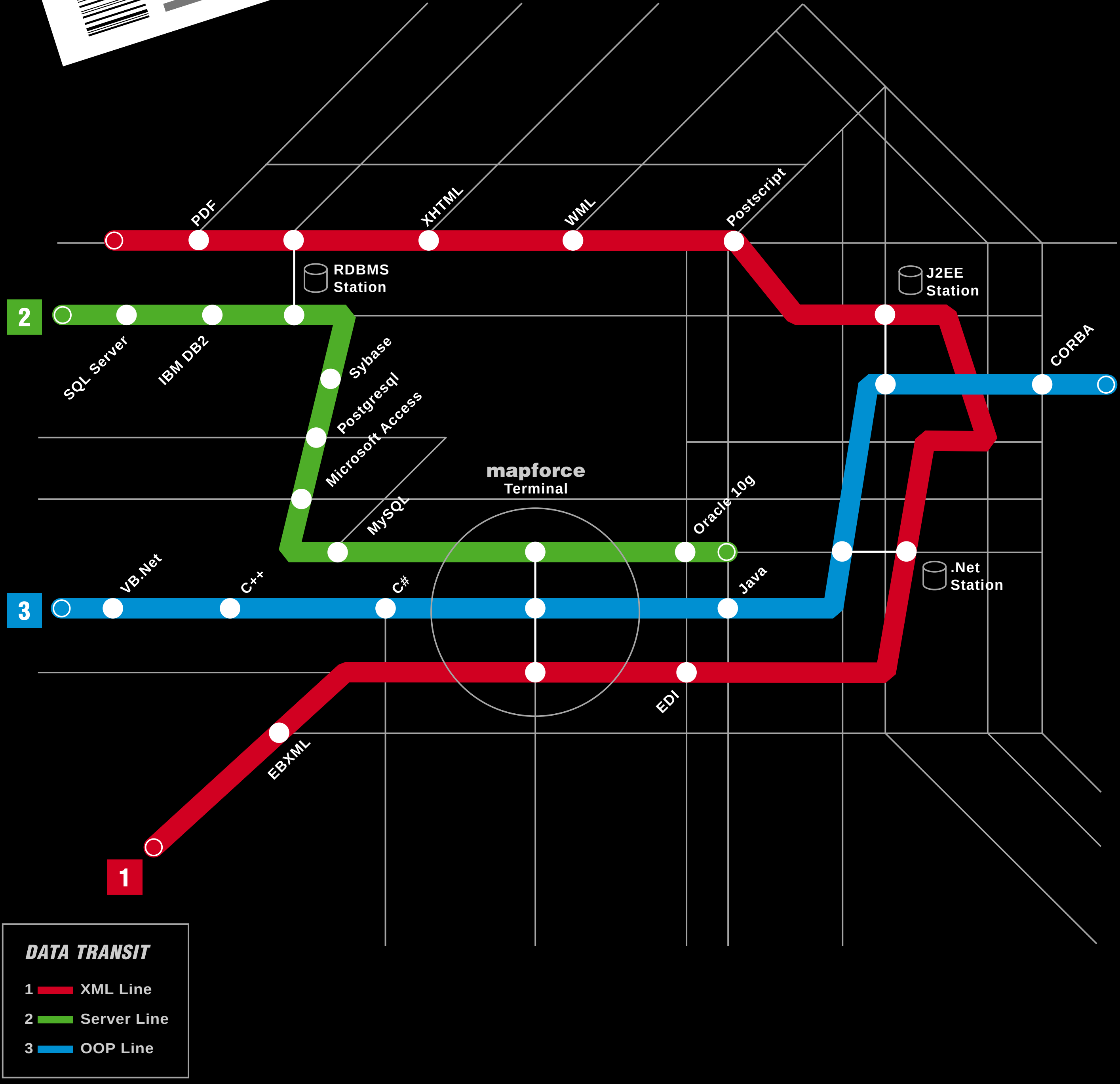
- *Java Community Process:* www.jcp.org/jsr/detail/168.jsp
- Abdelnur, Alejandro, et al. "Portlet Specification" Proposed Final Draft 2, Version 1.0, 08 August 2003.
- *OASIS Web Services for Remote Portal Web Site*: www.oasis-open.org/committees/wsrp
- Kropp, Alan, et al. " Web Services for Remote Portlets Specification". Approved as an OASIS Standard August 2003..

About the Author

Rickland Hollar is a senior applications architect with the Central Intelligence Agency with over 30 years of experience in the industry. Prior to joining the CIA, he was president of a Virginia-based software development firm.

■■■ rick_hollar@yahoo.com

Next Stop, Mapforce 2004!



Introducing Mapforce 2004! A powerful new XML mapping tool from Altova, producer of the industry standard XML Development Environment, XMLSPY. Mapforce is a visual data mapping tool, which auto-generates custom mapping code in multiple output languages such as XSLT, C++, C#, and Java, to enable programmatic XML-to-XML or XML-to-Database data transformations. Mapforce 2004 is the hassle-free transit for moving data from one format to another, enabling efficient information reuse. ***Download a free trial now!***

ALTOVA®
www.altova.com

Portals and Web Services

When business issues are technical

We've all heard the terms: portals, gadgets, portlets, dashboarding. But what does it all mean? And what role do Web services play in this exciting new world of componentized content?

WRITTEN BY
ALEC GRAZIANO

I would define "portalization" as the creation of an environment that gives the end user one-stop shopping for actionable content and collaboration. Portals don't create anything new; they merely give us comprehensive, personalized access to content that may reside in any number of repositories (see Figure 1). Moreover, portal environments benefit developers and implementers by providing a robust set of "out of the box" tools and features to ease application development and deployment. These tools include single sign-on (SSO) access to virtually any authentication data source, centralized authorization, logging and reporting, document versioning, and so on.

Content and Web services

In the vast expanse of where content may come from, Web services is just one of the vast sources that provide content. The difference, however, between Web services and other content-providing mechanisms is that Web services are a natural and synergic fit with portal environments. From the dawn of Web services time, the goal has been to deliver content in a standard, XML-based format that allows the consumers of that data to render or use it as they choose. Since personalization, localization, and customization are all standard fare in a portal environment, the presentation layer can be left to the developer of the portlet consuming the data. Sounds like the perfect job for good ol' Web services, doesn't it? It's also no secret that the great benefits to using Web services are their reuse and client-agnostic and protocol-independent properties. Given all this, the use of Web services in a portal environment is a no-brainer.

When to 'Portalize'

Does portalizing the weather along with a link to your company's homepage give you a good ROI on your portal software? I would guess probably not. But many other applications may make sense. Let's say we are a Web-engineering firm and at any given time our CEO wants to get:

- Information on sales leads that are in the pipeline
- Updates on development efforts that are in progress and status of client deliverables
- Meeting information for the day
- Access to the latest proposals with impending deadlines
- And maybe even an employee vacation schedule

Access to this content in a centralized, concise, easy-to-use format is a powerful thing. It eliminates the need to open and swap among multiple applications. It lets him log in to one SSO system and have access to all authorized resources rather than needing to log in to multiple systems. Furthermore, it eliminates the hassle of crawling through a document server's directory structure to find a specific proposal document. Finally, it allows him to collaborate with other team members on documents, meetings, etc. With its ability to improve project management and accelerate proposal and development efforts, the ROI for this type of portal application is very valuable, indeed.

The portal above can be described as a "dashboard" specific to our CEO's needs. It gives a personalized set of relevant information to the current end user. The information is a collection of data subsets from larger repositories. Those repositories in this case are a sales system, a project system, a scheduling system, a file structure, and an HR system. To create a portlet on top of an entire repository that exposes the complete functionality of that system may seem great, but in reality the end user wants only a slice of that data customized for them. For example, at first glance exposing the entire sales system sounds like it could be meaningful for many employees. However, the sales team is only interested in lead-flow information and not billing information, while the admin department only needs to see billing information so they can generate invoices. In an ideal portal application, each of these data slices remain separate as part of a specific user's personalized dashboard.

How Is It Done?

So how do we go about building a dashboard that is relevant and useful to each end user? This is done by leveraging a combination of built-in portal environment functionality and custom development. The portal environment gives the developer a nice framework for doing common tasks such as authentication, authorization, content formatting, and portlet behavior. Custom development helps to provide exposure and access to application content and functionality and makes the portal "consumable" by existing and future applications. The former development effort is very specific to the business need. If the business need allows, the content access exposure is best developed using Web services. By using Web services to expose key business functionality you are "future proofing" your application. This service may be consumed by a portlet today but may need to be consumed by a third-party application tomorrow. Once the necessary content and/or functionality are exposed, it must be consumed. This is the job of a portlet.

Today developers are somewhat limited when building consuming portlets. They must adhere to the APIs defined by the portal environment. The portability of these portlets is not quite there yet as standards like JSR 168 are still being worked out. If you do expose your content through Web services, the development effort to consume that data can be quite trivial in most portal environments. Most have simple wizard interfaces that allow a developer to point at a Web service and have the portlet nearly build itself. Due to the reusability, ease of deployment, and "future proofing" of your applications, using Web services to expose functionality and content is a smart move.

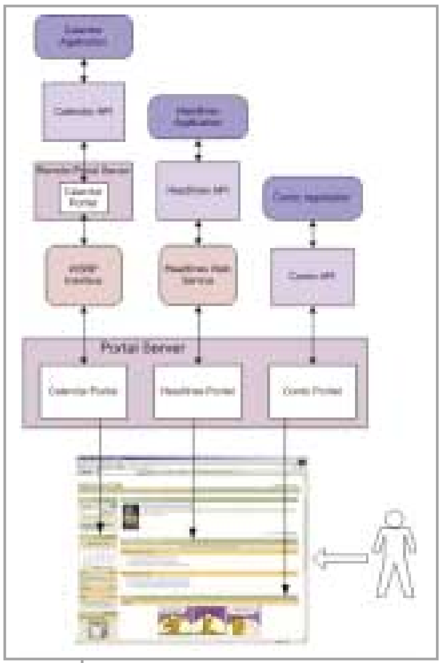


FIGURE 1 | **A typical portal environment**

SHOP ONLINE AT JDJSTORE.COM FOR BEST PRICES OR CALL YOUR ORDER IN AT 1-888-303-5282

BUY THOUSANDS OF PRODUCTS AT GUARANTEED LOWEST PRICES!

GUARANTEED BEST PRICES FOR ALL YOUR WEB SERVICES SOFTWARE NEEDS

JDJStore.COM

N-ARY
$299.00 — n-ary Ticket System v2.0

This installable application is a clean-cut Web-based tool that enables you to log & track important information. It can be used as an intranet or extranet product, giving your customers access to see how their issue is progressing. A unique number ID is assigned to every ticket. The system is extremely flexible and simple to use and can be utilized in numerous different situations. With more features than before, The Ticket System can be used for any number of great applications, which you can tailor as much or as little as you like.

GREENPOINT INC.
$1,350.00 — WEB CHARTS 3D VER. 4.7

WebCharts 3D is a state-of-the-art visualization package designed for the professional Web developer. It provides general purpose and specialized 2- and 3-dimensional charts, grids, and heat maps that can be delivered as server-generated interactive images (PNG, GIF, JPG, SWF, SVG, WBMP) or applets to browsers and mobile devices.

CHUTNEY TECHNOLOGIES
$755.25 — Chutney SOAP+ Toolkit

The Chutney SOAP+ Toolkit is the industry's first Web services monitoring and optimization toolkit. The Toolkit fills the functionality gaps in the leading SOAP development libraries by providing the ability to accurately pinpoint Web services bottlenecks and eliminate them through optimization techniques such as caching. The Chutney SOAP+ Toolkit acts purely as a supplement to these libraries, so no changes to the existing application logic are required. Flexible in its feature set, the Toolkit provides value to applications serving as either Web service consumers or providers.

ZION SOFTWARE
$995.00 — JAlerts' Deployment (JMessageServer)

If you need to send real-time Instant Messaging Alerts and Notifications to your employees or customers over the public IM services such as AIM, MSN, ICQ and Yahoo than JAlerts' is your solution.

JALERTS
$1,495.00 — JAlerts Deployment (ICQ)

If you need to send real-time Instant Messaging Alerts and Notifications to your employees or customers over the public IM services such as AIM, MSN, ICQ and Yahoo than JAlerts' is your solution. Based on Zion Software's popular JBuddy SDK technology, JAlerts' provides an even simpler solution for sending real-time messages.

INFRAGISTICS
$495.00 — Net Advantage Suite 2003 Volume 2

The Most Comprehensive Collection of Components for All Microsoft Development Environments. The ONLY fully integrated suite you'll ever need to create the most flexible, advanced applications for any Microsoft environment.

WWW.JDJSTORE.COM

OFFERS SUBJECT TO CHANGE WITHOUT NOTICE

Portal Environment: Build or Buy?

While today's portal software is impressive, it comes at a price. Portal software with implementation can run a company anywhere from the mid–five figures to well into the high six figures and beyond, depending on the number of users, servers, etc. Is it worth the investment to get into a portal environment? The answer really depends on your business need. Giving your staff access to their vacation day reports and links to HR policies within a branded intranet site may not be worthy of the sexy and expensive features of a full-blown portal environment. In this case, your efforts would be better spent handing this project to a junior developer and giving him or her a month to bang it out, giving you far greater ROI than deploying a real portal. If, however, you truly intend to portalize your business functions as described in our CEO dashboard example, there is no doubt you'll want to buy versus building your own. The time and investment needed to match the features and functionality of a portal environment are not time and money well spent. In this instance, a good portal environment is worth the investment.

While all portal environments differ in the goodies they offer, most will include some flavor of SSO authentication and authorization; the ability to define groups of users into communities with common interests and the associated content and applications; monitoring, management and reporting; content aggregation, indexing and searching; document check in, check out, and versioning; collaboration; portlet development APIs or SDKs; and much more. Since portal environments have been around for some time now, and most offer easy ways to develop portlets to their environment, giant libraries of portlets have been developed. Plumtree, for example, has hundreds of portlets in its library, from integrating with Lotus Notes, SAP, Siebel, PeopleSoft, and others, to using AOL Instant Messenger. Portal environment libraries provide a huge advantage to building all these components as many of them are open source or relatively inexpensive.

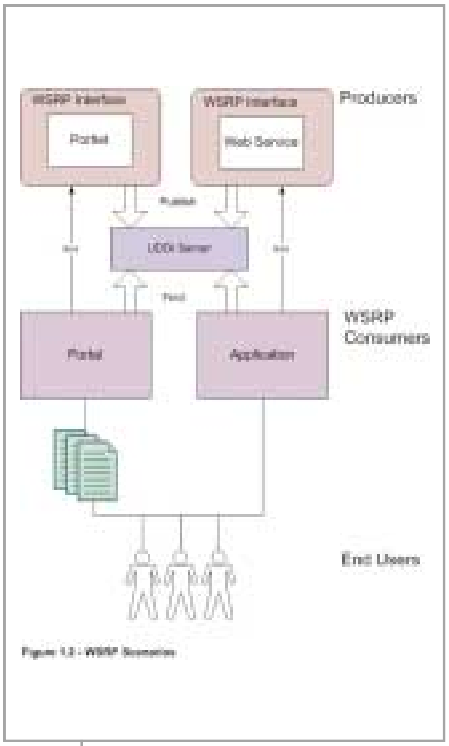


FIGURE 2 High-level view of two WSRP scenarios

> "It's also no secret that the great benefits to using Web services are their reuse and client-agnostic and protocol-independent properties"

What About Integration?

How well a portal environment integrates with Web services is also very dependent on the portal company you choose. While there are certainly some environments that make the integration process less painful than others, this should not be the developers' main consideration. Developers should be more concerned about how well a portal vendor interoperates with other vendors and portlets. In other words, how good is this vendor's support for emerging portal and Web service standards? At the beginning of the portal revolution, it was acceptable for each vendor to operate in its own box. Each portlet developed was proprietary to the environment in which it lived, as standards specific to portlets did not yet exist. Enter OASIS and the Java Community Process (JCP). With the recent approval of the Web Services for Remote Portlets (WSRP) standard by OASIS and the development of the JSR 168 Portlet Specification by the JCP, vendors now have the choice of whether they want to play nice together or not. Happily, several vendors have actually integrated these standards into their products already. This is a big step in the right direction and great news for those wishing to implement Web services as portlets.

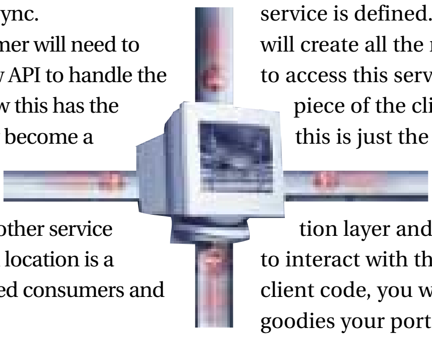
Web Services Standards and Portals

The WSRP is an interesting standard and somewhat of a paradigm shift from what we are used to in the Web services world. Until now everything was very data-centric. It's the nature of Web services to not worry about specific protocols or client displays. Web service calls either performed a function or they accessed some data. Presentation was left in the hands of the consumer. The WSRP standard has changed this for portlets. The idea is that no longer are portlet Web services "data-oriented," as OASIS calls it. They are now "presentation-oriented" (see Figure 2).

WSRP defines a Web service interface that allows a much richer interaction with Web services than straight Web service calls. The producer of the WSRP-compliant service can expose description information and capabilities of the service, presentation markup that is now part of the service, an optional registration interface for creating a relationship between the consumer and producer, and optional management interfaces. By creating WSRP-compliant portlets, producers are able to publish portlets that have been developed in their familiar environments and be assured that consumers will receive the service as it was intended from the application logic up to the presentation. It lets them reuse portlets, make them WSRP compliant, and expose them to other consumers. From the consumer side, no additional development is required to integrate a WSRP portlet, as there would be if a straight Web service were being consumed. So long as the consumer is also WSRP compliant, the inclusion of this portlet is virtually development free.

UDDI service so new consumers of the service may develop to the new API. This is okay for new consumers, but your existing consumers are now out of sync.

Each existing consumer will need to develop against this new API to handle the fax data. You can see how this has the potential to very quickly become a maintenance nightmare. If controlling presentation, data, and other service attributes from a central location is a requirement due to varied consumers and



“…in reality there are still some limitations to seamless, portable, simple Web service portlet integrations”

The adoption of the WSRP standard will ultimately make portlet interoperability issues between portal vendors a problem of the past, opening up new opportunities for businesses to expose and market their developed portlets while maintaining full control over application and presentation logic. It will also benefit businesses looking to consume existing WSRP portlets by offering them a complete packaged portlet application, presentation and all, requiring virtually no development effort.

Though it is clear that there are many benefits to using Web services in a portal environment, there are still some drawbacks. Until the portal standards have been fully adopted, developers will need to continue to produce and consume Web services the “old fashioned” way. Consuming this way will require a good deal of additional development to create presentation and logic around the data you get back. Another problem with standard Web services is that you are constrained by the APIs you build. Let’s say, for example, that you decide your fancy sales lead Web service should now return fax numbers, which it didn’t do before. You will need to first produce an updated Web service containing this new field. Then this definition WSDL needs to be republished into a

ease of deployment, you may want to choose a vendor who is on board with these new portal standards.

Implementing Web Services

So, let’s say you’re willing to take the good with the bad and want in on using data-oriented Web services in a portal, how do you go about doing that? Like everything else, this is dependent on the environment you are in. But here are the basics you will need to follow: first you must develop the business logic that needs to be exposed. Since we are in the Web services world you can do this with any language and on any platform you like. Next, you need to expose this business logic as a Web service. At this stage of the game just about every IDE has some support for generating the files you need for Web service exposure. These files may include your Web service wrapper code, WSDL definition file, deployment descriptors, etc. Deploy this service and publish its description to a UDDI server and you have successfully produced a Web service.

Now you must consume this service in your portal. The development kits that come with your portal software will determine how easily you integrate your newly deployed service. If you’re lucky, you’ll have a wizard-style interface that will ask you where the service is defined. From the WSDL, the wizard will create all the necessary client proxy files to access this service. This is still only one piece of the client work. In the Java world this is just the M of the MVC (Model-View-Controller). You will still need to build the presentation layer and any logic that is necessary to interact with the service. In building the client code, you will have access to all the goodies your portal environment offers, such as session information, portlet-to-portlet communication, authorization information, and so on. Implementing portlets as Web services will take some work. How smoothly your implementation goes will depend on what development platform you use, what portal vendor you choose, and what in-house talents you can leverage to handle this development.

Conclusion

Individually Web services and portals are very powerful technologies. One would think then that together they would be unstoppable. In theory that’s correct, but in reality there are still some limitations to seamless, portable, simple Web service portlet integration. The good news is that these are known problems and are being addressed by both standards bodies and the portal vendors themselves. If your business needs dictate it, a portal can be a wise investment. Your ROI will not take that long to be seen if your portal is used properly. And when all the portlet libraries have been searched with no luck for that special application you require, take a stab at creating your portlet with a Web service. For all the current and future standards, platform and protocol independence, reusability, interoperability, and “future proofing” benefits you get, you’ll be glad you did.

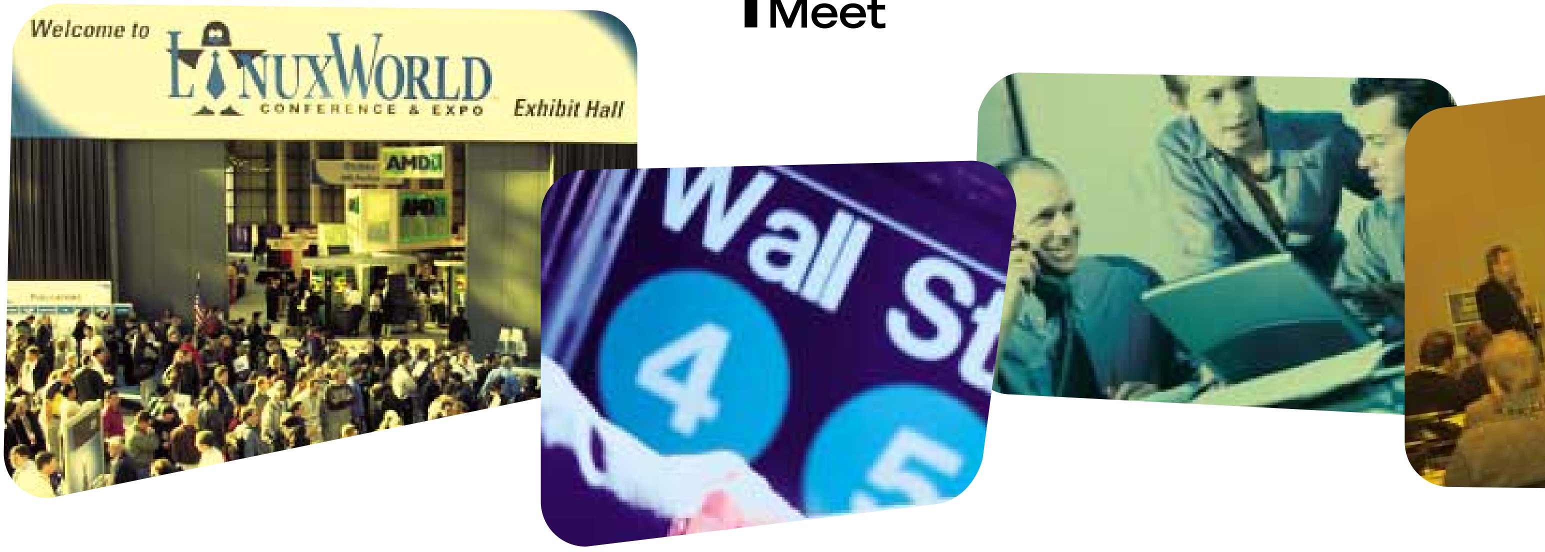
About the Author

Director of Web engineering at Miller Systems, Alec Graziano is responsible for all of Miller Systems’ Internet-based software development engagements and process methodology, including design, architecture, and programming. Alec has significant and broad experience in designing and managing the delivery of industry-standard, Web-based software, particularly in the Web services arena, in both J2EE and .NET frameworks.

agraziano@millersystems.com

LinuxWorld® Conference & Expo

Where Open Minds Meet



Keynote Speakers

Chris Stone
Vice Chairman, Office of the CEO, Novell, Inc.
Wednesday, January 21
9:15 am – 10:00 am

Dave Dargo
Vice President, Linux Program Office, Oracle
Wednesday, January 21
11:45 am – 12:30 pm

Tom Killalea
Vice President of Infrastructure, Amazon.com
Wednesday, January 21
2:45 pm – 3:30 pm

Sam Greenblatt
Senior Vice President and Chief Architect, Computer Associates Intl.
Thursday, January 22
11:30 am – 12:15 am

Ross A. Mauri
General Manager, e-Business on demand, IBM Systems Group
Thursday, January 22
1:45 pm – 2:30 pm

CONFERENCE: January 20 – 23, 2004

EXPO: January 21 – 23, 2004

THE JAVITS CENTER: New York, NY

LinuxWorld Conference & Expo is the world's leading and most comprehensive event focusing on Linux and open source solutions. This January, discover how companies across the globe have achieved higher profits and increased their productivity by utilizing Linux, the fastest growing operating system in the world.

- Get in-depth coverage on the latest Linux and open source developments and learn about innovative product solutions from the industry's top companies.
- Network with the leaders in the open source movement and discuss with your peers how to best leverage the technology for your organization.
- Participate in LinuxWorld's world-class education program and benefit from interactive training in the all-new Hands-on Labs!
- Attend the 2nd annual Linux Financial Summit and hear how Wall Street firms are leveraging Linux to achieve a lower total cost of ownership. (Sponsored by BEA)
- Discover real world practical solutions and find answers to today's most critical IT issues.

Cornerstone Sponsor

ORACLE®

Platinum Sponsors

Computer Associates® IBM intel hp invent Novell. Sun microsystems SuSE simply change

REGISTER ONLINE WITH REFERENCE CODE: A-WJD

www.linuxworldexpo.com

IDG WORLD EXPO

Fiorano ESB

A strong enterprise service bus solution

Web services have staked their claim as a key technology in building and integrating large, distributed enterprise systems. More often than not, however, Web services may be just one piece of a myriad of interfaces. Not only are IT workers faced with working against heterogeneous interfaces, but process management, workflow, administration, and security are also important. That's where Fiorano ESB (Enterprise Service Bus) comes into play.

WRITTEN BY
BRIAN BARBASH

Fiorano ESB is, to use their term, a brokered peer-to-peer system providing an integration and services infrastructure based on standard protocols. Participants in the ESB are part of an asynchronous, event-driven system based on messaging technology.

As seen in Figure 1 from the Fiorano Web site, several components make up the ESB. Peer Servers are the distributed elements of the system that may reside on any number of machines throughout a network. They host and run the individual enterprise services that make up a full Fiorano ESB. As the name implies, Peer Servers may communicate directly with one another creating a peer-to-peer network. Peer Servers also have the option of communicating over a central messaging pipe that may be any JMS-compliant messaging system. The Super Peer is the administrative hub of the entire ESB application, providing monitoring services, security, and configuration management and application development and deployment capabilities.

For the purposes of this review, I'll focus on the Fiorano Business Service Composer. This product provides the capability to create, orchestrate, and manage components within the ESB, including Web Services.

Fiorano Business Service Composer

The Business Service Composer is an environment in which multiple independent services are pulled together as part of a larger business process. The Service Composer ships with several prebuilt services that may be used out of the box:

- Services for managing process flow
- Adapters for various external systems including SAP, databases, file systems and Web services
- Bridging services providing access to mail systems via SMTP, Microsoft's MSMQ, and Enterprise JavaBeans

For this review, I'll use a couple of simple Web services that will participate in a larger business process. The first is a Personal Information service, built with .NET and hosted locally, that accepts a person's name and returns their address and telephone number. The second service is a reference to the Unisys public weather service that accepts a zip code and returns the local forecast.

Pulling Things Together

Building a business process in the Service Composer is easy. As seen in Figure 2, services are represented by icons in the workspace and connected by data flow lines. These data flow lines represent the connections between individual enterprise services. These connections may be configured as point-to-point or publish/subscribe.

The SOAP adapter is used to configure each Web service in the Fiorano ESB application. As you would expect, the SOAP adapter is based on the WSDL definition of the target Web service. The WSDL may be hard-coded into the adapter, read from an existing URL, or retrieved from a UDDI registry. Once the WSDL has been captured by the adapter, the developer must choose which of the Web service's operations to use in the application. Each selected operation will be represented by a pair of input and output points on the service icon in the Service Composer. The next step to configuring the adapter is to set up the authorization mechanism. The adapter supports interaction with the Web service through a proxy and may be configured to respond to a Basic HTTP Authentication challenge. It also supports using Web services over HTTPS. The adapter can then be configured to retry service calls in case of failure. Parameters available for resubmitted requests include the total number of attempts, the interval between attempts, and the timeout period for a request. Finally, a DTD is presented that represents the output of the Web service. The developer may edit the DTD to correct any errors or add missing elements.

Company Info
Corporate Headquarters
Fiorano Software, Inc.
718 University Avenue, Suite 212
Los Gatos, CA 95032 U.S.A.
Tel. +1.408.354.3210
1.800.663.3621 (Toll-Free within the USA)
E-mail: info@fiorano.com

Licensing Information
CPU-based licensing for enterprise and peer servers; standard, end user–based tools licensing

Requirements
Any platform that runs a JVM; no application server needed. Tested platforms include Windows, Solaris, AIX, HP-UX, Linux, AS400, OS390, and others.

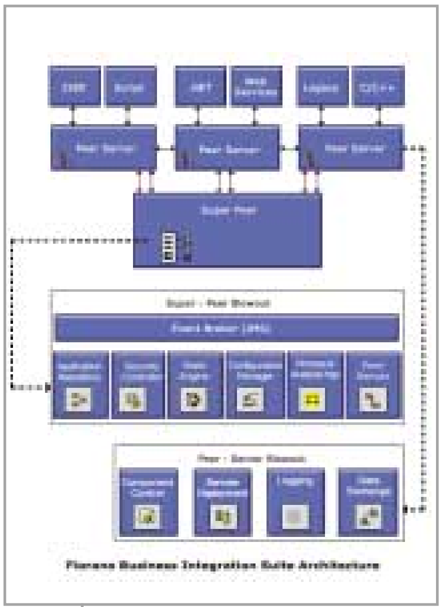


FIGURE 1 **Fiorano ESB Architecture**

All data flowing through any Fiorano ESB application is in XML format, with each service defining its unique input and output signatures. The Fiorano Mapper provides a visual mapping tool to transform the XML documents to and from the various signatures. Figure 3 shows an instance of the Mapper that handles the transformation of the results of the Personal Information service to the input of the Unisys weather service. To establish a mapping, simply drag the elements from the input service on the left to the workspace for the desired output node. The Mapper provides an extensive set of functions to apply to the data during transformation, including string operations, control flow, and data aggregation among others. The end result of the exercise is an XSL style sheet that is applied to the data as it moves between service nodes.

The Fiorano Worklist

One interesting component of the Fiorano ESB is the Worklist service, which provides a general container for the XML documents in a business process. When data is posted to the Worklist, the information is held pending a notification to release it for downstream processing. Documents in the Worklist may be searched, modified, or deleted. The Worklist is a Web service based on the Apache Axis product. Therefore, custom applications may be developed to interact and monitor each worklist in a business process.

Debugging

While an application is running, developers may insert breakpoints on any connection between services. Once data reaches a breakpoint, the value may be inspected and easily modified. Messages may then either be forwarded for downstream processing or removed from the system with no further operations performed.

Another useful tool in the debugging process is the Display service. The Display service may be the target of any service functions as a general output. Messages posted to a Display may be viewed in their raw data format or in hierarchical format. Message headers and attachments are also available for inspection.

Monitoring and Managing Systems

Fiorano ESB applications and their constituent services may be directly started and stopped from within the Service Composer. Fiorano ESB also provides the capability to modify applications while they are executing. In the example I have set up, while the system was processing data I added a new data feed to repeatedly send in requests to the Personal Information service. Once the feeder was defined, the Service Composer deployed and started the service without interruption to the existing processes.

Summary

Enterprise applications present unique challenges when integrating them into larger business processes. Developers and IT workers must deal with heterogeneous interfaces, workflow, process management, security, and administration. Fiorano's Enterprise Service Bus is one alternative for stitching these systems together. Their Business Service Composer is an easy-to-use IDE for orchestrating services, mapping data, and leveraging Web services into business processes and is a strong asset in the enterprise service bus solution.

About the Author

Brian R. Barbash is the product review editor for *Web Services Journal*. He is a consultant for the Consulting Group of Computer Sciences Corporation, where he specializes in application architecture and development, business and technical analysis, and Web design.

■■■ bbarbash@sys-con.com

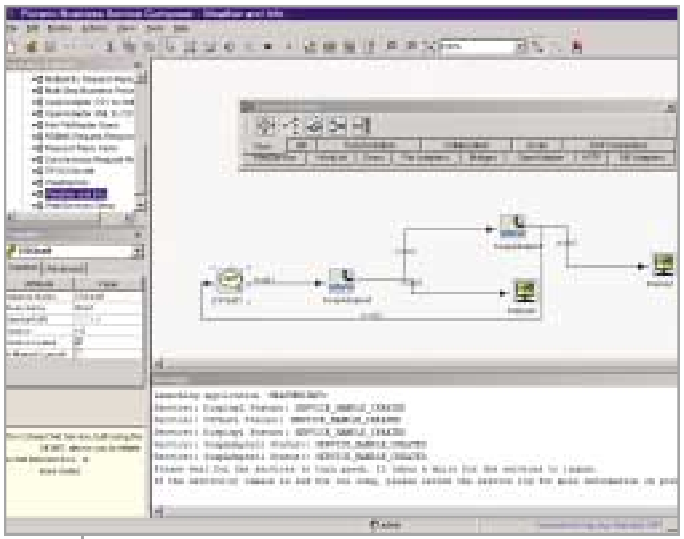


FIGURE 2 **Fiorano Business Service Composer**

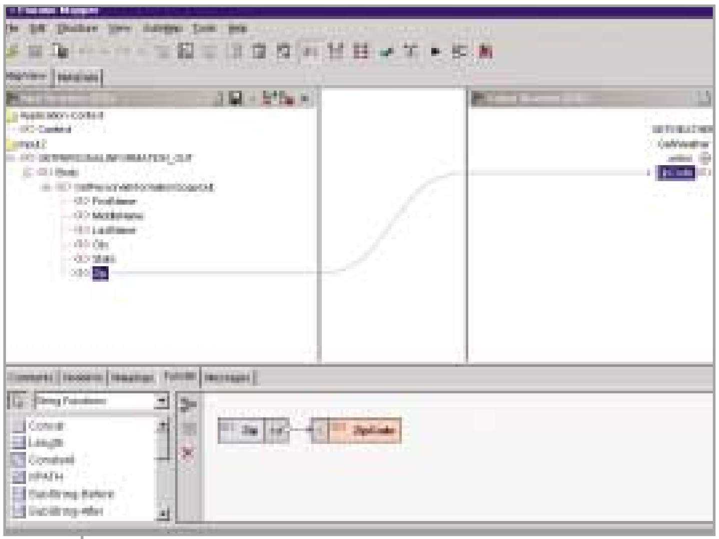
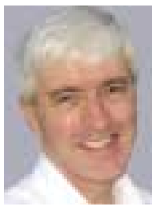


FIGURE 3 **The Fiorano Mapper**

Using Web Services for Business, part II

Making yourself understood

If the content of a SOAP message is not understood or the recipient of a message does not know what to do with it when they get it, then using Web services for business, even with extensions for reliable delivery and security, will just not work.



WRITTEN BY
DAVID BURDETT

To solve this problem you not only need to define how to "deliver the message" (see ***WSJ***, Volume 3, Issue 11), you also need to define what the contents of a message mean as well as define what the recipient should do with a message when it is received. In other words, you need to define the message semantics.

This brings me to the purpose of this article, which is to describe the different semantic definitions you need to create so that Web services can be used for business. It then describes the leading standards that are being developed to support the definition of semantics and suggests an architectural approach to solving the problem of handling multiple different semantic definitions.

Why Standard Semantic Definitions Are Needed

Standard semantic definitions are needed for three main reasons:

1. ***Shared understanding:*** Without a precise definition of what things mean that is shared and agreed to by the developers and implementers of the systems being built and connected using Web services, there will be confusion and no interoperability.
2. ***More choice:*** If the semantics of services, messages, business documents, and so on, are all defined with the same definition language, then it should matter less what technology was used to develop them, enabling more customer choice in the tools they use.
3. ***Reduced information loss:*** If a buyer and a supplier each create their own versions of the order documents they create or accept, but they are different, then the buyer might create data that the supplier can't use and the supplier might be missing some information that they need.

For example, what does a customer ID in an order mean? Is it the identifier the supplier uses to identify the customer, or is it the one used by the customer to identify itself? Similarly, if an invoice is sent to a business it could mean "please pay the amount specified in the invoice for services and products provided" but it could just as easily mean "please calculate the tax and customs duties due on this invoice, then return the invoice so that I can send it to the customer". Without a clear statement of what the Web service that receives a message should do with the message, interoperability will not occur.

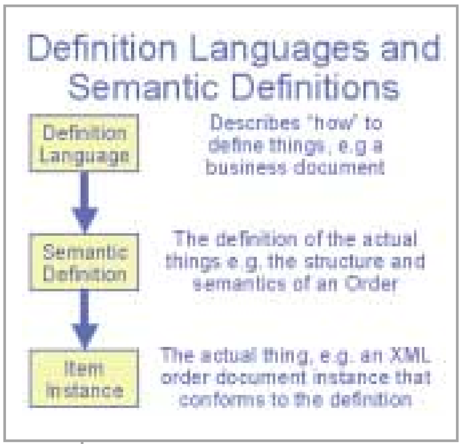

Definition Languages and Semantic Definitions
Definition Language
Describes "how" to define things, e.g a business document
Semantic Definition
The definition of the actual things e.g. the structure and semantics of an Order
Item Instance
The actual thing, e.g. an XML order document instance that conforms to the definition

FIGURE 1 Definition languages and semantic definitions

The Role of Context

Semantic definitions also vary depending on the context. For example, a date is always a date and identifies a particular day in a year. However, the meaning of a date can vary depending on where the item is being used. For example, a "shipment date" is the date on which a shipment is made and a "sell by date" is the date by which a product should be sold.

Even this can be further qualified depending on the purpose for which the data is being used. For example, in a planning document a shipment date could be a date that identifies when the buyer is requesting the goods be shipped. Later, in an Advanced Shipment Notice, it would be the date set by the supplier that identifies when the supplier plans to deliver the goods; later still. In a Delivery Note, it would be the actual date the goods were shipped.

This means that semantic definitions must almost always include information about the context in which the definitions should be used.

What Semantic Definitions Are Needed?

Creating semantic definitions requires two different types of standards: *Definition Languages*, which describe "how" you can describe the semantics and structure; and *Semantic Definitions*, which define the "actual" structure and semantics (see Figure 1). For example, XML is a "definition language" that can be used to define business documents such as an order or invoice. However, to get interoperability you also need to create a "Semantic Definition" for an actual order explaining what each field means as well as creating XML definitions that define what the order looks like on the wire.

In general, two groups of semantic definitions need to be created using these standards: data-related, including languages to define business documents, messages, and transformations or maps; and process-related, including languages to define business processes, choreographies, and service interfaces (see Figure 2).

Note: the remainder of this article references several standards initiatives. Brief descriptions and links to these initiatives, together with a glossary of the terms used are available from the *Web Services Journal* Web site at www.sys-con.com/webservices/sourcec

WebAppCabaret™

J2EE Web Hosting

www.webappcabaret.com

Quality Web Hosting at a reasonable price...

<JAVA WEB HOSTING AND OUTSOURCING>

You have developed the coolest mission-critical application. Now you need to deploy it.
Outsource your hosting and infrastructure requirements with WebAppCabaret so you can save time and money and concentrate on other important things. WebAppCabaret is the leading **JAVA J2EE** Web Hosting Service Provider. From shared hosting to complex multi-dedicated server hosting, WebAppCabaret has the right solution for you. **30 Day Money Back Guarantee and SLA**.WebAppCabaret offers the latest Standards based Servlet containers, EJB servers, and JVMs. We provide options such as e-Commerce, **EJB 2.x**, Failover, and Clustering. Our **Tier 1** Data Center ensures the best in availability and performance.

At WebAppCabaret you have the flexibility to choose the LATEST hosting technology best suited for your WEB application or your programming skill. If you are a programmer or consultant,WebAppCabaret has the right hosting solution for your project or client's dynamic web application/services requirements.

OUTSOURCING:
Do you really need an IT department for your web applications, mail systems, and data backups when we can perform the same functions more efficiently at a fraction of the cost - with competent technical expertise and redundant hosting facilities.

J2EE HOSTING:
Below is a partial price list of our standard hosting plans. (**Reseller accounts also available**). For more details please log on to **http://www.webappcabaret.com/jdj.jsp**

Features	Basic	Professional	Enterprise	Dedicated
Monthly Cost	us$9.00	us$17.00	us$39.00	us$191.00
Servlet Spec	Latest	Latest	Latest	Latest
JSP Spec	Latest	Latest	Latest	Latest
EJB Spec			Latest	Latest
Private JVM	x	x	x	x
Database	1	1	5	Unlimited
e-Commerce		x	x	x
Tier 1 Center	x	x	x	x
Accounts/Server	120	120	35	1
RAM/Server	2GB	2GB	2GB	256MB
Max Memory Heap	20 MB	30 MB	60 MB	256MB
RAID Diskspace	60MB	200MB	900MB	40GB
Bandwidth/Month	3GB	10GB	20GB	55GB
Backup	x	x	x	x
Domains	1	2	5	Unlimited
Email Accounts	2	20	100	Unlimited
Servlet Contexts	1	4	20	Unlimited
Control Panel	x	x	x	x
FTP	x	x	x	x
Telnet/SSH		x	x	x
Web Mail	x	x	x	x
Web Stats		x	x	x
Perl/PHP		x	x + ASP.NET	x + ASP.NET
Dedicated IP			x	x
Engine Choices	x	x	x	x
WAR/EAR	x	x	x	x
Tomcat		Latest	Latest	Latest
JBoss			Latest	Latest
Jetty	Latest	Latest	Latest	Latest

Prices, Plans, and Terms subject to change without notice. Please log on to our website for the latest Prices and Terms.

.cfm. (Further analysis of Web services developments is also available on the author's weblog at www.davidburdett.com)

The Semantic Web

The Semantic Web is a W3C effort that describes how to use RDF to define additional information about resources on the Web so that they can be more easily understood by machines. Although the Semantic Web will make searching for resources on the Web more accurate, it isn't very helpful when using Web services for business as developers (i.e. people), rather than automated software, will, for the foreseeable future need to understand the semantics so that they can work out how to use Web services to connect businesses together.

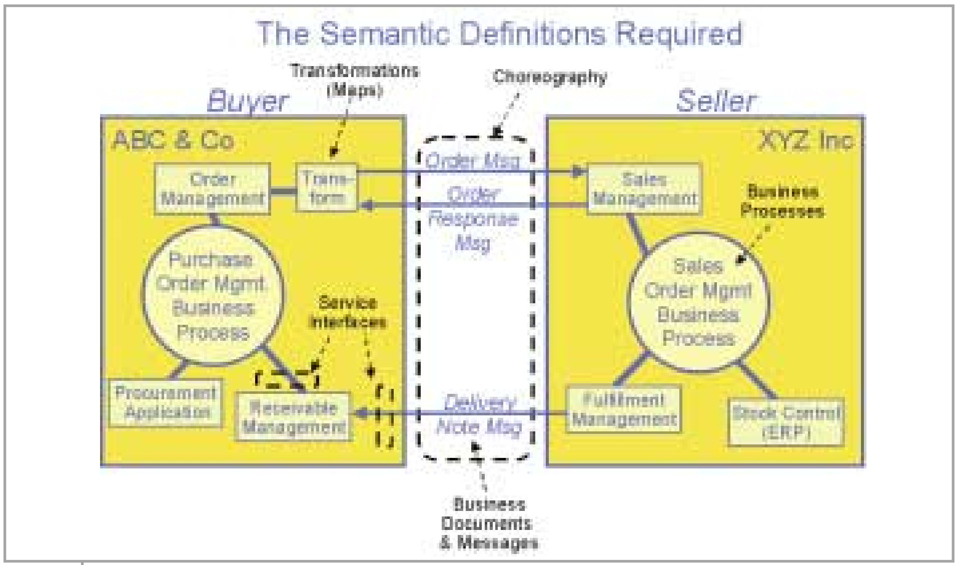


FIGURE 2 | **Required semantic definitions**

Data-Related Semantics

Data-related semantics describe the business documents and messages that are exchanged between Web services. As the sender and receiver of the message may use different message and document formats, transformations or maps are also needed to translate between them. Each of these areas is reviewed in turn.

Business Document Definitions

Unsurprisingly, XML is "the" definition language for defining business documents. Because of its popularity and ease of use, many organizations – RosettaNet, PIDX, ACORD, and hundreds of others – have created their own business documents and semantic definitions using XML.

This variety of business documents, often with slightly varying semantics and different XML representations, means that at least in the short term transformations have to be used by either the sender or the receiver. However, transformations require technology to make them work. This means that some businesses – especially smaller businesses that work in different industries – won't be able to support all the different formats they would like because of the costs involved.

The initiative that promises to address this problem most directly is Universal Business Language (UBL) from OASIS, which is developing a set of simple supply-chain business documents that are universally applicable. It's gaining support from many vertical industry groups who plan to use it as the basis for versions in their industries. Other popular business document formats include OAGI and xCBL.

Message Definitions

A message consists of one or more business documents wrapped inside a SOAP message – either with or without attachments – with additional SOAP headers for security, reliable delivery, etc. (see November's article).

WSDL allows messages to be defined in the context of a particular service. However, messages can be reused by many services, by different businesses, and in many different contexts (see earlier). It would be better if messages were defined separately from the service definition so that a service definition could just reference the message definition. However, there are no standards efforts to develop a separate definition language for messages.

Lower implementation costs would also result if semantic definitions that specified the structure and semantics of a specific combination of business documents, the version of SOAP, and associated SOAP Headers for actual messages were standardized. However, no standardization efforts are attempting this mainly because of the wide variety of business document standards; the need to finalize standards for security, reliable delivery; and the absence of a stand-alone standard for defining messages.

That said, a number of vertical industry initiatives, such as RosettaNet, PIDX, CIDX and others, have created their own message definitions that are either non-SOAP or that use SOAP-based specifications such as ebXML Messaging.

The result is that there is still a long way to go before standardized message definitions will be available.

Transformation (Maps)

The lack of standardized business documents and messages means that implementers will often need to transform the data they handle, requiring technology to do the actual transformation and "maps" that define what the transformation should do.

The only existing "standard" for transformations is the powerful and flexible XSLT. However, this power makes it less than ideal for use in business document transformations as most current XSLT implementations store the complete XML document in memory. This limits XSLT usefulness since business documents can sometimes be large – at least several megabytes.

There are also few transformation maps publicly available that describe how to map between the popular document formats as most maps are built into proprietary mapping software.

Until standard business document formats become widely accepted, a business can minimize costs by adopting an "internal" business document standard, e.g. xCBL or UBL, to which business documents received from other systems and partners are mapped. This means that instead of having to create a transformation map for each combination of business document and message format you use, you only need to create maps to and from your standard formats – significantly reducing the number of maps required.

Process-Related Semantics

Process-related semantics are the next area where standard definitions are needed. These definitions include (see Figure 2): internal *business processes definitions* that typically execute on either a single system or within a single business; *choreography definitions* that describe how two or more independent businesses or processes cooperate in terms of the sequence of the messages they exchange; and *service interface definitions* that describe the interfaces to a service that is either executed by a business process or is a target of one of the messages in a choreography definition.

The distinction between a business process definition and a choreography definition is important. A business process definition describes how internal, private business processes work – for example the Sales Order Management process where a business uses its sales management system, stock management system, and fulfillment system to satisfy orders that the business receives. In this case, the business handling those orders is in complete control of how those internal and external systems are integrated and combined with existing manual processes.

Choreography definitions, on the other hand, define how one "independent" business or process interacts with another by defining the sequence and conditions in which messages are exchanged between them. In this latter case no single business or process is in control so each has to agree with the other on how to cooperate. For example, if a buyer sends a supplier an order, the supplier needs to know how to respond. Should they: a) return an order response indicating the extent to which they can meet the order, b) just ship the goods and send an invoice, or c) do something different. No single business can unilaterally decide what do without informing, and getting the agreement of, the other businesses involved.

Let's look at these in more detail.

Business Process Definitions

As I said earlier, business processes by their very nature tend to focus on operations within a single business or organization. Good, effective business processes are also one of the ways in which businesses realize competitive advantage. One business is unlikely, therefore, to want to or need to share its internal business process definitions with anyone else.

Historically, specifications such as XLANG, WSFL, BPML, and others have provided ways of defining a business process by linking together Web services. However, the Business Process Execution Language for Web Services (BPEL) is now replacing them as the leading standard in this area. All these languages enable new business processes to be defined by combining Web services that have been defined using WSDL.Once a new business process is defined, access to the functions it provides may itself be exposed as a Web service by providing a WSDL definition for it.

Given that business process definitions won't often be shared outside of an individual business and therefore are not critical to interoperability, why will languages such as BPEL be used? There are probably two reasons: consultants and others with expertise in a particular business topic or vertical industry may develop process definitions that represent "best business practice," which they then sell to their customers as the basis for their client's solutions; and a business may develop a business process definition that represents their own "best practice," which they can then replicate in several divisions of their business.

Although porting business process definitions between platforms should be possible, it is likely to face the same problems as porting between Java implementations – it's not too hard as long as you haven't used any platform-specific extensions.

Choreography Definitions

Choreography definitions, on the other hand, are critical to interoperability as they define how two or more businesses or processes cooperate by specifying the sequence and conditions in which messages are exchanged together with explanations of what sending a particular message means.

This is important, at least in the short term, as a Web service will often be used as a "front-end interface" for some existing "legacy" application with hard-to-change behavior that will process the data. If the semantics of the choreography is not both well defined and properly understood by each role involved (e.g., buyer and seller), then misunderstandings will often lead to incorrect processing of the message.

At run time, choreography definitions, as long as you know the role you are taking, can be used to verify that the messages are exchanged in the sequence required. Also, since a single definition is shared between all the roles, it is easy to independently check prior to "going live" that each implementation behaves correctly. Checking the compatibility of separate, independently designed business process definitions that businesses do not want to share is harder, especially if several roles are involved.

Major choreography definition language standards include BPSS, which focuses primarily on B2Bi, is closely tied to other ebXML specifications, and not easily used with other Web services specifications; and WS-Choreography, which is developing a choreography definition language that can be used to define how businesses or Web services in general can cooperate.

Several vertical industry groups, such as RosettaNet, have defined their own choreography definitions. However there are no cross-industry initiatives working on standard choreography definitions that would be more widely applicable.

Web Service Definitions

WSDL version 1.1 is mature in the way that it defines an interface to a Web service in terms of the input and output messages, and the destinations (called ports) to which messages are sent. The WSDL1.2 specification is also being developed and is making good progress, although there are many differences with version 1.1, making it unlikely that it will be backwards compatible with WSDL 1.1.

However, more work is needed on both specifications so that they can record additional information such as the choreography definitions the service supports as well as additional SOAP features such as security and reliable delivery.

Although none are being developed, standardized service definitions with clearly defined semantics for sending/receiving a message would help as there would then be a standard way of connecting businesses, for example to place an order, where only the URL of the destination and security information needed to be changed.

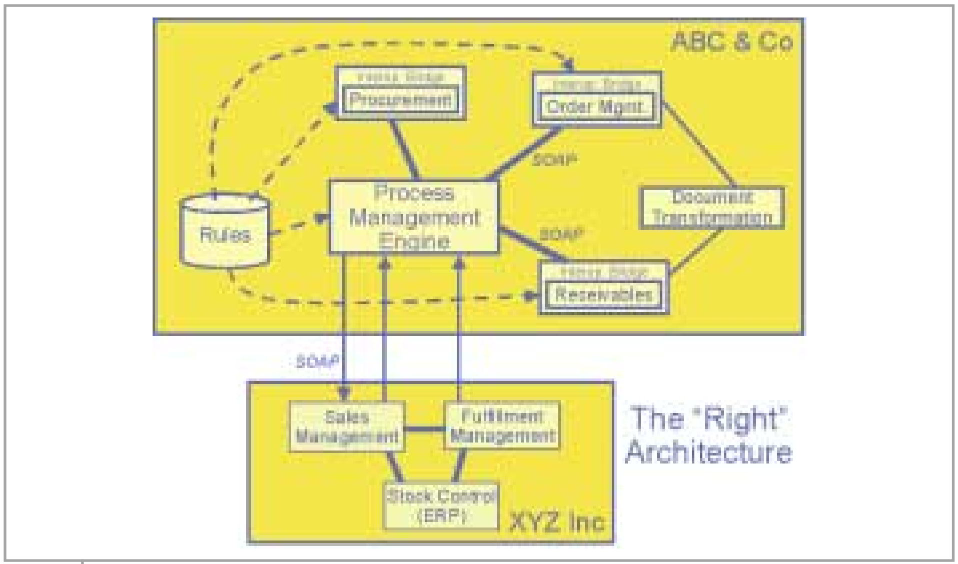


FIGURE 3 **The "right" architecture**

What Should You Do Now?

Clearly there are multiple standards, and in particular, multiple different semantic definitions for business documents, service interfaces, and choreographies. This means that, in the short term, a business will have to support multiple different ways of connecting to their existing systems and partners. However, waiting for standards to mature so that connectivity is easier will mean missing the real benefits of better, lower-cost solutions that using Web services for business can bring.

Adopting the Right Architectural Approach

The solution to this problem is adopting the "right" architectural approach (see Figure 3) for the solutions you build or buy. This approach, which is an extension of the approach discussed in last month's article, solves the problem by expanding the "interoperability bridge" to transform business documents as well as handle different Web services standards. It also includes a "process management engine" that dynamically selects the business process definitions to use that support the choreography definition required.

It is also a good idea to adopt a single "internal" business document standard (described earlier) to simplify transformations of business documents as well as expose internal applications and the external systems you connect to as small, discrete Web services that provide a single function such as accepting an order. As a result it is much easier for the process management engine to link those Web services together in different sequences so that you can more easily and rapidly support the wide variety of different choreography definitions that many businesses will be asked to support.

This approach separates the business application from the changing environment in which it operates. As a result you can, relatively independently, change an application, the business process, the business documents, and the Web service standards being used. It also means that when changes occur the interdependencies can be handled in one place – in the rules – at lower cost and more consistently.

About the Author

David Burdett is director of Standards Strategy, Web Services for Commerce One and is responsible for directing their use of Web services standards. He is also an experienced consultant with over 20 years' experience in managing and motivating teams in IT strategy, development, architecture, and organization.

■■■ david.burdett@commerceone.com

JAVA: INDUSTRY NEWSLETTER
JAVA DEVELOPERS JOURNAL Industry >Newsletter
aligo
JAVA DUDES
JDJ NEWS DESK EXCLUSIVE

WEB SERVICES: INDUSTRY NEWSLETTER
WebServices JOURNAL Industry >Newsletter
ORACLE
ONLINE EXCLUSIVE: FULL "KEYNOTE PANEL" TRANSCRIPT

THE INSIDER INTELLIGENCE YOU NEED...

TO KEEP AHEAD OF THE CURVE

FREE E-Newsletters

SIGN UP TODAY!

Go to www.SYS-CON.com

The most innovative products, new releases, interviews, industry developments, and plenty of solid *i*-technology news can be found in SYS-CON Media's Industry Newsletters. Targeted to meet your professional needs, each e-mail is informative, insightful, and to the point. They're free, and your subscription is just a mouse-click away at www.sys-con.com.

SELECT THE INDUSTRY NEWSLETTERS THAT MATCH YOUR NEEDS!
CHOOSE ONE – OR TRY THEM ALL!

JAVA DEVELOPERS JOURNAL Industry >Newsletter	WebLogic Industry >Newsletter
WebServices Industry >Newsletter	WebSphere Industry >Newsletter
XML JOURNAL Industry >Newsletter	ColdFusion Industry >Newsletter
wireless Industry >Newsletter	.NET Journal >Newsletter

Don't Delay! Subscribe for FREE! at www.sys-con.com

Exclusively from the World's Leading *i*-Technology Publisher SYS-CON MEDIA

Pragmatic Web Services Security Today

Simple strategies for securing and monitoring Web services

Concerns about security are cited as the single largest barrier to rapid Web services adoption. Yet most Web services today are fairly straightforward point-to-point integrations that can be securely implemented using only digital certificates and the Secure Sockets Layer (SSL) protocol.

WRITTEN BY
JOTHY ROSENBERG

Regardless of security strategy, enterprises are well advised to monitor their Web services to ensure security has not been compromised. Taken together, widely available standard security technologies and active monitoring provide a sensible approach to the majority of today's Web service security challenges. This article describes how to use these technologies to secure the most common deployments of Web services quickly and easily. I'll close with a brief introduction of WS-Security and how this emerging standard relates to what you do and do not get with SSL.

Web Services Security in Perspective

When considering what will be needed to enable a ubiquitous service-oriented architecture (SOA) in the next 3–5 years, the security challenge looks daunting. The WS-Security standards that specify security infrastructure that will allow the safe delegation of trust and identity are still evolving. The maturing of these standards is a necessary step toward the realization of a true service-oriented architecture. However, if we step back and focus on how Web services are really being utilized today, we find that practical, secure deployments are possible now. In contrast to the fully distributed applications of the future SOA "nirvana," most of today's Web services are simple point-to-point integrations.

As the name suggests, Web services can be seen as a logical evolution of the previous generation of distributed computing – the World Wide Web. It should thus come as no surprise that much of the security infrastructure developed for the browser-based Web is directly applicable to the realm of server-to-server Web services. Indeed, the combination of well-known Internet security technologies and best practices for monitoring security compliance are the primary requirements for securing today's initial Web service deployments.

Web-enabled business-to-consumer (B2C) commerce provided much of the impetus for the development of the SSL protocol and digital certificates. B2C commerce required confidentiality and integrity for the communications between a consumer running a browser-based client, and the front-end application server responsible for facilitating the transaction. Authentication of the server was also critically important to ensure that the consumer could trust the party requesting personal and confidential information such as credit card numbers.

In the case of Web services, the need for confidentiality and integrity are similar, but because the primary communications are server-to-server, the requirement for authentication is truly symmetric. In addition, new monitoring requirements arise because Web services applications are capable of exposing strategic business assets. Information sharing provides leverage and business advantage to partners as long as that communication is secure. As enterprises make their information assets more widely accessible internally and externally they must be sure that strategic information is available only to authorized parties.

Securing Point-to-Point Web Services

As I stated earlier, the most common external Web services being implemented today are point-to-point. The key security requirements for this type of Web service are authentication and confidentiality. Authentication ensures that the Web service client and server are known to each other. Confidentiality ensures that data exchanged between the parties are kept secret and intact. In some cases, integrations within the firewall must be similarly secured due to concerns about the critical nature of the assets accessed through a Web service, or regulatory requirements. In fact, Gartner has found that $1trillion has been lost by corporations due to theft of company information. These security requirements can be met using proven and familiar technology, namely the SSL protocol and digital certificates.

Consider a fictional retailer, Gargantuan Books, which would like to offer its resellers the ability to check stock and pricing using Web services. The security of Gargantuan's systems and data is critical to the success of the Web services initiative. Gargantuan needs to know which resellers are accessing which Web services and that communications between Gargantuan and its resellers are being kept confidential.

The StockCheck and PriceCheck Services

Gargantuan decides to provide all its resellers with the ability to check book availability via the StockCheck Web service. Gargantuan's gold-level resellers will be allowed to check both availability and pricing via a special Web service, the PriceCheck service. Both services take a unique identifier for the book (its ISBN) along with the number of copies being requested; they return a stock status of OnStock, Partial, or BackOrder. The PriceCheck service also returns the price to this reseller for the book. Resellers plan to use these services to provide store clerks and online customers with book availability and pricing information.

Gargantuan creates these Web services and provides its resellers with the WSDL description shown in Listing 1.

Securing the Services

At this point Gargantuan's Web services are *not secure*! Any application with access to the Internet could call them at

```
http://www.gargantuanbooks.com/ws/Resell
er.asmx
```

and check Gargantuan's stock and pricing. In order to secure them, Gargantuan can activate mutual (aka symmetric or "2-way") SSL on the Web servers exposing these services. Using SSL involves (1) procuring a digital certificate for the computer running the Web service, (2) toggling the SSL handshake configuration setting on the Web server to require client authentication, and (3) procuring digital certificates for the computers that will be the clients of the Web services (see sidebar, "How SSL Works").

Procuring and Enabling Digital Certificates

Digital certificates for SSL can be acquired quickly, simply, and cheaply from various public Certificate Authorities (CAs) such as VeriSign, GeoTrust, and Comodo. A wizard on the Web server guides the entire process from initial application to final X.509 certificate install on the computer. Once installed on both ends of the communication, the SSL request-response protocol for two-way authentication is fully automatic for Web services, just as it is for one-way, browser-server interactions.

So, Gargantuan procures a digital certificate for the server running the Web service, sets the SSL handshake configuration setting on the server to require client authentication, and informs its resellers that they must procure digital certificates for the computers that will be clients of these Web services.

Gargantuan now safely provides these services at

```
https://www.gargantuanbooks.com/ws/Resel
ler.asmx
```

(Note the https: prefix, that 's' indicates SSL.)

How Secure Are StockCheck and PriceCheck?

Gargantuan has met most of its security requirements by setting up mutual SSL. Not only does the server authenticate itself to clients but clients are required to authenticate themselves to the server as well. Clients without valid certificates cannot access the service and communications between Gargantuan and its resellers are protected by strong encryption (see sidebar, "Procuring and Enabling Digital Certificates").

This is an adequate level of protection for many Web services; however, as with any security scheme, abuses can still occur. Gargantuan needs to monitor its Web services in order to detect and respond to abuses. Just as banks need surveillance cameras alongside hardened vaults and security guards, Gargantuan needs an active monitoring solu-

How SSL Works

The most widely deployed and used system of security on the Internet is Secure Sockets Layer (SSL). SSL is what makes the padlock symbol light up in your browser when you go to a secure site. It is used not just for security between a server and a browser; it can also be used between two servers. SSL provides three important security capabilities: server authentication to the client, confidentiality of the transmitted data, and (optionally) client authentication to the server.

Server authentication ensures that the domain the client expects to be visiting is indeed where the server is. This authentication happens through a digital certificate installed on the server. The certificate has a domain contained in it that must match the actual domain of the server.

Confidentiality of transmitted data is provided by the activation of the SSL protocol. This protocol – built into all common Web and application servers – does an initial secret key exchange and cryptographic protocol negotiation; that protocol is then used to encrypt all data transmitted between client and server. No one has been able to demonstrate any vulnerability of SSL encrypted data on the wire.

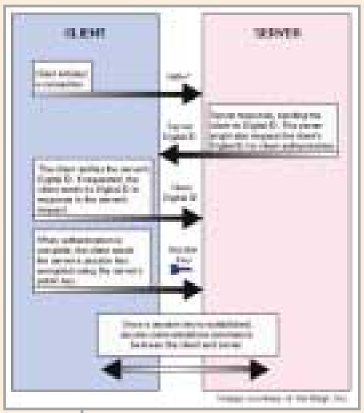


FIGURE 1 How SSL Works

Procuring and Enabling Digital Certificates

Using the built-in IIS Server Certificate Wizard, it's easy to generate the key pair that is needed to submit a request to a Certificate Authority for a server digital certificate. A base-64 encoded certificate signing request (CSR) is saved as a text file representing all the information provided. It is this text file (which contains the public key just generated) that will be submitted to the certificate authority for the creation of the X.509 digital certificate which will enable SSL.

Using the standard enrollment page of one of the public certificate authorities (in this case FreeSSL), the CSR from the aforementioned text file is copied and pasted into the Web form. This provides the CA with all the necessary information to validate the identity of the submitter and the rights of that submitter to the specified domain. (That's all that's needed to acquire an SSL certificate.) The validation and certificate generation process takes anywhere from 10 minutes to four days depending on the CA and the type of certificate requested.

Once the X.509 certificate is generated, it is delivered by the certificate authority as a base-64 encoded text file. The IIS Certificate Wizard is invoked again, this time for the purpose of installing the certificate.

The text file containing the certificate is provided to the wizard, which decodes and parses the certificate and presents the information back to the administrator for confirmation purposes. Upon completion of this step, the certificate is installed; it will now enable SSL connections to be made either by browser clients or by servers acting as clients.

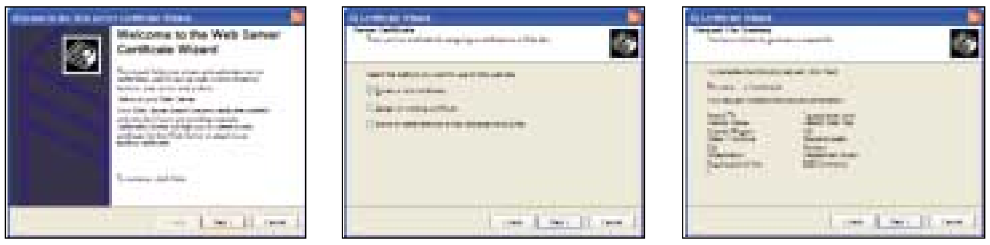


FIGURE 2 | SSL enrollment

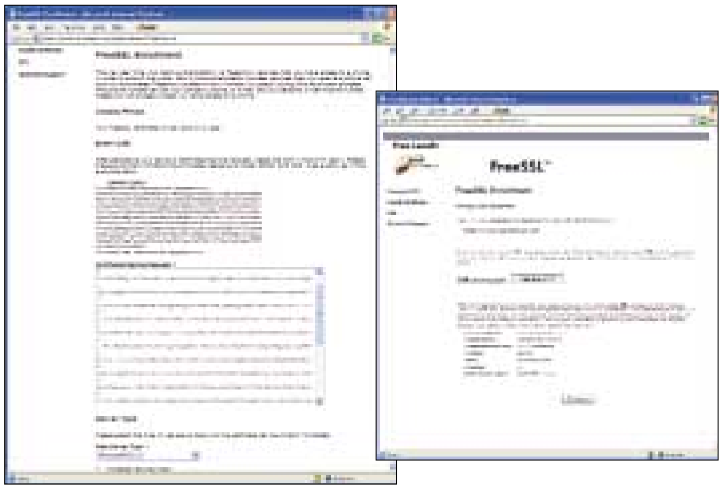
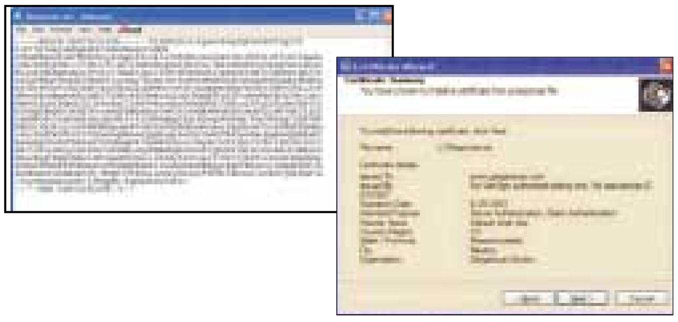


FIGURE 3 | Enabled certificate



tion to monitor the performance of its security measures and to provide key information in the event that there is a security issue. Active monitoring is essential regardless of the security approach used.

Monitoring and Enforcing Security

In order to monitor security, Gargantuan needs to be able to "see" the identity of the clients accessing each of its Web services. It is not enough just to know that a certain client accessed the server; Gargantuan needs to know that only Gold authorized resellers are accessing the PriceCheck service. To demonstrate how this can be done, we'll continue our example using a specialized Web services management tool to monitor security for Gargantuan Books.

A powerful Web services monitoring tool enables companies to quickly isolate and resolve run-time issues, including security problems. Typically, a tap or sensor is placed at each Web service end-point in the flow of messages that monitors all Web services. With this highly scalable and low-overhead architecture, all service requests and responses are completely visible. We can monitor the security of all Web services, alert operators of security problems, and record detailed information for use in resolving security issues.

Web Services Monitoring Is Different

Due to the self-describing nature of Web services thanks to WSDL and XML Schemas, Web services monitoring tools can discover what Web services are running by reading all appropriate WSDL files. In this way the tools can then present monitoring information using the language defined by the Web services themselves.

When a Web service client first accesses a Web service, the monitoring tool detects the "establish session message", which contains the identity of the client from the SSL handshake. It can then monitor and analyze all elements of the Web service stream, including client identity. It logs the information for later analysis and generates alerts to service operators and systems management applications. In this case, Gargantuan wants to log the identities of all clients attempting access, alert a security officer when unauthorized clients attempt to access services, and record detailed information on all unsuccessful access attempts.

Gargantuan can not only monitor and enforce its security policies, but can also monitor the performance and availability of its Web services, diagnose service problems, and gain real-time visibility into the business activity flowing through its Web services network (see sidebar "Monitoring Web Services Security").

How Secure Is StockCheck Now?

The combination of secure communications via SSL and active monitoring has secured Gargantuan's point-to-point Web services. The services are secure – Gargantuan and its resellers are authenticated – and all communications are protected by strong encryption. An active monitoring solution is essential to ensure that security is maintained: all accesses are logged for later analysis and failed access attempts generate security alerts. As an added bonus, Gargantuan is able to address other operational challenges and gain business insight from its Web services stream.

Beyond SSL-Secured Point-to-Point Web Services

The security story for Web services by no means stops here. As we progress beyond today's point-to-point Web services, SSL will not be enough. We will need message-level security. The building blocks for message-level security are XML Signature and XML Encryption.

XML Signature supports the security principles of integrity and nonrepudiation. XML Signature is a W3C standard that can be placed inside an XML document or it can refer to external elements that are signed. It is important to be able to just sign part of an XML document, such as when patient information is updated by an authorized health care provider. XML Signature might refer to external elements when it is being used to guarantee integrity of critical Web pages to prevent defacement.

XML Encryption supports the security principles of confidentiality in transit as well as persistent confidentiality when messages are stored (something SSL definitively can not do). This is a critical need when maintaining adherence to security policies and regulatory compliance. As with XML Signature, XML Encryption can be

Monitoring Web Services Security

Using an active monitoring solution, Gargantuan can monitor the security of its Web services, alert security officers of problems, and log information in order to resolve security issues.

The tool auto-discovers Gargantuan's Web services and exposes the Web service operations and fields for monitoring and analysis (see Figure 4). The screen capture on the right shows the main configuration screen. Gargantuan uses this graphical interface to create a security dashboard, and rules for alert generation and logging.

Gargantuan has requested monitoring all HTTP and HTTPS handshakes including client-side SSL authentication. SSL is initially turned off, clients are not authenticated, and the services are unsecured. SSL is activated at 2:31 between the fourth and fifth entry in the log. The log shows an attempt to connect without client authentication which results in a 403 http failure. It also shows client authentication information for all accesses after SSL is activated.

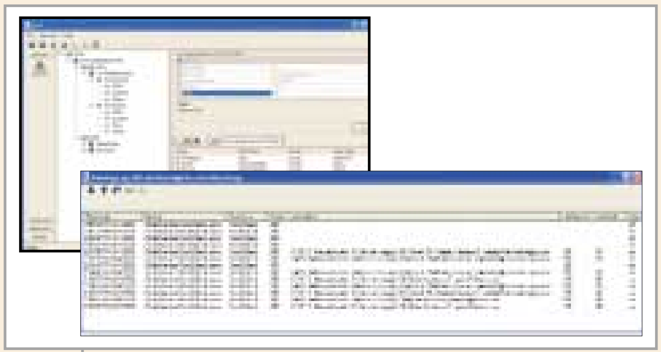


FIGURE 4 Monitoring SSL Authentications

applied to just a portion of an XML document, such as the credit card number being transmitted with an order.

WS-Security builds on the solid security base W3C established with XML-based security. Just as XML Signature and XML Encryption are about XML security, WS-Security is about SOAP security. It is a set of extensions to SOAP. Its role is to specify how to pass security information in SOAP headers.

IBM, Microsoft, and VeriSign developed and submitted WS-Security to OASIS. The OASIS Web Services Security (WSS) technical committee is now driving the standard effort forward with a focus on using XML Security with Web services. The simplest way to think about WS-Security is that it defines security tokens passed in the SOAP header. The most important tokens passed in this manner are for authentication and authorization. Why? Because this specifies who signed the XML message that is in this SOAP message's payload. Or it specifies what the individual is allowed to do ("purchases up to $10,000") that might not be consistent with the Purchase Order carried in the payload.

The typical way WS-Security will work is that all or part of an XML message will be signed or encrypted and in the SOAP header will be passed a signature or decryption key to be used by the message recipient.

As with any security system, the keys to its success are the policies it is designed to enforce and the monitoring of how effective it is at such enforcement. Therefore, it is critical that all authentications, all authorizations, all linkages between identities and the transactions they perform, and the regulations affecting encrypted XML data elements all be monitored and logged for discovery and defense.

Conclusion

Point-to-point Web services can be securely implemented today. The industry-standard Web security technologies SSL and digital certificates form the basis for simple and effective Web services security. Properly implemented, they provide authentication and ensure communication confidentiality and integrity. Coming soon are ways to apply message-level security that will allow Web services to handle sensitive information that must remain encrypted and Web services that take multiple hops. In order to ensure security no matter what type of technology is applied, companies must also monitor security compliance at run time. SSL, digital certificates, and an active security monitoring system provide a sensible approach to securing the majority of today's Web services when used together.

About the Author

Dr. Jothy Rosenberg is a serial entrepreneur and the co-founder, CTO, and CEO of Service Integrity, a rapidly growing Web services monitoring and analysis software provider. Prior to this venture, Jothy cofounded GeoTrust, the world's second largest certificate authority. He is also the author of *How Debuggers Work* and the forthcoming *Building Secure Web Services with WS-Security* (Sams; 2004).

■■■ jothy@serviceintegrity.com

Listing 1: WSDL for Stockcheck and PriceCheck

```
<?xml version="1.0" encoding="utf-8"?>
<definitions name="Reseller"

targetNamespace="http://ws.gargantuan.com:83/Reseller.xml"
  xmlns:tns="http://ws.gargantuan.com:83/Reseller.xml"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/" >

  <message name="StockCheckReq">
    <part name="ISBN" type="xsd:string"/>
    <part name="Quantity" type="xsd:long"/>
  </message>
  <message name="StockCheckResp">
    <part name="Status" type="xsd:string"/>
  </message>
  <message name="PriceCheckReq">
    <part name="ISBN" type="xsd:string"/>
    <part name="Quantity" type="xsd:long"/>
  </message>
  <message name="PriceCheckResp">
    <part name="Price" type="xsd:decimal"/>
    <part name="Status" type="xsd:string"/>
  </message>

  <portType name="ResellerPort">
    <operation name="StockCheck">
    <input message="tns:StockCheckReq"/>
    <output message="tns:StockCheckResp" />
    </operation>

    <operation name="PriceCheck">
    <input message="tns:PriceCheckReq"/>
    <output message="tns:PriceCheckResp" />
    </operation>
  </portType>

  <binding name="ResellerBinding"
     type="tns:ResellerPort">
    <soap:binding style="rpc"
     transport="http://schemas.xmlsoap.org/soap/http"/>

    <operation name="StockCheck">
      <soap:operation
      soapAction=

"http://ws.gargantuan.com:83/Reseller.xml#StockCheck"/>
      <input>
        <soap:body use="encoded"
          namespace=
            "http://ws.gargantuan.com:83/Reseller.xml"
          encodingStyle=
            "http://schemas.xmlsoap.org/soap/encod-
ing/"/>
      </input>
      <output>

        <soap:body use="encoded"

namespace="http://ws.gargantuan.com:83/Reseller.xml"
          encodingStyle=
            "http://schemas.xmlsoap.org/soap/encoding/"/>
      </output>
    </operation>

    <operation name="PriceCheck">
      <soap:operation
      soapAction=

"http://ws.gargantuan.com:83/Reseller.xml#PriceCheck"/>

      <input>
        <soap:body use="encoded"

namespace="http://ws.gargantuan.com:83/Reseller.xml"
        encodingStyle=
          "http://schemas.xmlsoap.org/soap/encoding/"/>
      </input>

      <output>
        <soap:body use="encoded"
          namespace=
            "http://ws.gargantuan.com:83/Reseller.xml"
          encodingStyle=
            "http://schemas.xmlsoap.org/soap/encoding/"/>
      </output>
    </operation>
   </binding>

   <service name="Reseller">
     <port name="ResellerPort"
binding="tns:ResellerBinding">
       <soap:address location=
          "http://www.gargantuan.com/ws/Reseller.asmx" />
     </port>
   </service>
</definitions>
```

Download the code at
sys-con.com/webservices

www.LinuxWorld.com

Security: Protect Your Linux Environment
LINUXWORLD MAGAZINE
On Location at the Render Farm
DreamWorks is ready to take Linux into new worlds
Linux Alternatives to Microsoft Windows Server Products

The **Leading Magazine** for Enterprise and IT Management

LinuxWorld Magazine

There is no escaping the penetration of Linux into the corporate world. Traditional models are being turned on their head as the open-for-everyone Linux bandwagon rolls forward.

Linux is an operating system that is traditionally held in the highest esteem by the hardcore or geek developers of the world. With its roots firmly seeded in the open-source model, Linux is very much born from the "if it's broke, then fix it yourself" attitude.

Major corporations including IBM, Oracle, Sun, and Dell have all committed significant resources and money to ensure their strategy for the future involves Linux. Linux has arrived at the boardroom.

Yet until now, no title has existed that explicitly addresses this new hunger for information from the corporate arena. *LinuxWorld Magazine* is aimed squarely at providing this group with the knowledge and background necessary to make decisions to utilize the Linux operating system.

Look for all the strategic information required to better inform the community on how powerful an alternative Linux can be. *LinuxWorld Magazine* does not feature low-level code snippets but focuses instead on the higher logistical level, providing advice on hardware, to software, through to the recruiting of trained personnel required to successfully deploy a Linux-based solution. Each month presents a different focus, allowing a detailed analysis of all the components that make up the greater Linux landscape.

Regular features include:

- *Advice on Linux Infrastructure*
- *Detailed Software Reviews*
- *Migration Advice*
- *Hardware Advice*
- *CEO Guest Editorials*
- *Recruiting/Certification Advice*
- *Latest News That Matters*
- *Case Studies*

SAVE 30% OFF!

REGULAR ANNUAL COVER PRICE $71.76

YOU PAY ONLY

$49.99

12 ISSUES/YR

**OFFER SUBJECT TO CHANGE WITHOUT NOTICE*

SUBSCRIBE TODAY!

WWW.SYS-CON.COM

OR CALL

1-888-303-5282

FOR ADVERTISING INFORMATION:

CALL 201 802.3020 OR VISIT WWW.SYS-CON.COM

Service-Oriented Integration: An Evolutionary Step for IT

Start small, and experiment

This article outlines a set of best practices for service-oriented integration (SOI) by reviewing the evolution of integration practices, applying those lessons to service-oriented architectures (SOA), and finally analyzing SOA and SOI with the specific technology set of Web services today and into the future.

WRITTEN BY
BILL DEFOREEST &

SCOTT ROSENBLOOM

Services and Integration: How Did We Get Here?

The advent of the information age has brought an increased emphasis on timely access to information. Use of the Internet to reach employees, customers, and business partners creates corresponding demands to deliver solutions through different kinds of interfaces in shorter time frames. These compressed time frames frequently eliminate some traditional options for satisfying new application requirements, such as migrating to a new, state-of-the-art packaged application; or creating a custom solution built in isolation from the existing enterprise systems.

Enterprise-level integration offers a potential solution. By leveraging existing assets and building out only the required new logic, integration can shorten time frames and costs while providing the new interface access. The ongoing issue has been the difficulty in actually achieving the potential. While there are many different ways to view mainstream integration practices, we think it is useful to divide it into three main categories:

- ***Custom-coded projects:*** For consultants or IT staff, composing a custom solution is potentially viable on a small scale. Over time and scale, this approach has some obvious limitations. Many complex problems such as security, reliability, scalability, support for transactions and related semantics are solved and then maintained by each enterprise over and over again. Increasing complexity and decreased timelines are making this path risky. The savings in technology acquisition is being swamped by the cost of maintenance and time to delivery.
- ***Messaging middleware and integration brokers:*** Products such as IBM's WebSphere Business Integration Message Broker or webMethods' Integration Platform take the form of central "hub-and-spoke" messaging systems that transform and route messages based on custom sets of rules. The brokers are often packaged as suites with process automation tools and interfaces to popular databases and applications. These products have typically been built around proprietary technologies that were designed to solve data synchronization problems on a massive scale, resulting in hugely complex products with unfamiliar tools. The out-of-the-box interfaces in these products often struggle with higher level forms of integration that involve business logic, leading to unexpected custom coding. These two factors result in products that are very expensive and require many successful projects to show a return on investment.
- ***Application servers:*** J2EE implementations, Microsoft's .NET, and other platforms have been developed and marketed as robust and scalable platforms with standard sets of services such as security and transaction monitoring. Adapter frameworks and business process features allow a customer to build automated processes in addition to user-facing applications. A high degree of skill is needed to effectively leverage these features. Coupled with the need to design and code an entire solution from the ground up, achieving the best architectures and solutions still requires a substantial investment beyond technology acquisition.

Thus, many organizations find themselves with seemingly conflicting needs. On the one hand, incremental delivery of application functionality with a quick return on investment (ROI) must be provided in short time frames. On the other hand, a sustainable and manageable enterprise architecture must be developed that can support the business into the future. This is where service-oriented integration can provide some assistance (see Figure 1).

SOI Defined

In the most generic sense, a "service" can be defined as a piece of functionality that is transparently accessible over a network. As the underlying concept behind SOI, service-oriented architecture brings some refinement to the broad notion of a service:

- ***Components in an SOA ecosystem are often published:*** The clients that use them are not predefined. Theoretically, any client that understands the interface could utilize the functionality.
- ***The notion of multiple clients extends to an expectation of interoperability with multiple technology mediums:*** As an example, the service can be created using J2EE, an

A LIMITED TIME SAVINGS OFFER FROM SYS-CON MEDIA

SUBSCRIBE TODAY TO MULTIPLE MAGAZINES

AND SAVE UP TO $400 AND RECEIVE UP TO 3 FREE CDs!*

RECEIVE YOUR DIGITAL EDITION ACCESS CODE INSTANTLY
WITH YOUR PAID SUBSCRIPTIONS

3-Pack

Pick any 3 of our magazines and save up to $275⁰⁰
Pay only $175 for a 1 year subscription plus a **FREE CD**

- 2 Year – $299.00
- Canada/Mexico – $245.00
- International – $315.00

6-Pack

Pick any 6 of our magazines and save up to $350⁰⁰
Pay only $395 for a 1 year subscription plus **2 FREE CDs**

- 2 Year – $669.00
- Canada/Mexico – $555.00
- International – $710.00

9-Pack

Pick 9 of our magazines and save up to $400⁰⁰
Pay only $495 for a 1 year subscription plus **3 FREE CDs**

- 2 Year – $839.00
- Canada/Mexico – $695.00
- International – $890.00

CALL TODAY! 888-303-5282

Pick a 3-Pack, a 6-Pack or a 9-Pack

☐ 3-Pack	☐ 1YR	☐ 2YR	☐ U.S.	☐ Can/Mex	☐ Intl.
☐ 6-Pack	☐ 1YR	☐ 2YR	☐ U.S.	☐ Can/Mex	☐ Intl.
☐ 9-Pack	☐ 1YR	☐ 2YR	☐ U.S.	☐ Can/Mex	☐ Intl.

TO ORDER •Choose the Multi-Pack you want to order by checking next to it below. •Check the number of years you want to order. •Indicate your location by checking either U.S., Canada/Mexico or International. •Then choose which magazines you want to include with your Multi-Pack order.

☐ **MX Developer's Journal**

U.S. - Two Years (24) Cover: $143	You Pay: $49.99 /	Save: $167 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $29.99 /	Save: $60
Can/Mex - Two Years (24) $168	You Pay: $79.99 /	Save: $137 + FREE $198 CD
Can/Mex - One Year (12) $84	You Pay: $49.99 /	Save: $40
Int'l - Two Years (24) $216	You Pay: $89.99 /	Save: $127 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $59.99 /	Save: $30
Digital Edition - One Year (12)	You Pay: $19.99	

☐ **Linux World Magazine**

U.S. - Two Years (24) Cover: $143	You Pay: $79.99 /	Save: $63 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $39.99 /	Save: $32
Can/Mex - Two Years (24) $168	You Pay: $119.99 /	Save: $48 + FREE $198 CD
Can/Mex - One Year (12) $84	You Pay: $79.99 /	Save: $4
Int'l - Two Years (24) $216	You Pay: $176 /	Save: $40 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $99.99 /	Save: $8

☐ **Java Developer's Journal**

U.S. - Two Years (24) Cover: $144	You Pay: $89 /	Save: $55 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $49.99 /	Save: $22
Can/Mex - Two Years (24) $168	You Pay: $119.99 /	Save: $48 + FREE $198 CD
Can/Mex - One Year (12) $84	You Pay: $79.99 /	Save: $4
Int'l - Two Years (24) $216	You Pay: $176 /	Save: $40 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $99.99 /	Save: $8

☐ **Web Services Journal**

U.S.- Two Years (24) Cover: $168	You Pay: $99.99 /	Save: $68 + FREE $198 CD
U.S. - One Year (12) Cover: $84	You Pay: $69.99 /	Save: $14
Can/Mex - Two Years (24) $192	You Pay: $129 /	Save: $63 + FREE $198 CD
Can/Mex - One Year (12) $96	You Pay: $89.99 /	Save: $6
Int'l - Two Years (24) $216	You Pay: $170 /	Save: $46 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $99.99 /	Save: $8

☐ **.NET Developer's Journal**

U.S. - Two Years (24) Cover: $168	You Pay: $99.99 /	Save: $68 + FREE $198 CD
U.S. - One Year (12) Cover: $84	You Pay: $69.99 /	Save: $14
Can/Mex - Two Years (24) $192	You Pay: $129 /	Save: $63 + FREE $198 CD
Can/Mex - One Year (12) $96	You Pay: $89.99 /	Save: $6
Int'l - Two Years (24) $216	You Pay: $170 /	Save: $46 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $99.99 /	Save: $8

☐ **XML-Journal**

U.S. - Two Years (24) Cover: $168	You Pay: $99.99 /	Save: $68 + FREE $198 CD
U.S. - One Year (12) Cover: $84	You Pay: $69.99 /	Save: $14
Can/Mex - Two Years (24) $192	You Pay: $129 /	Save: $63 + FREE $198 CD
Can/Mex - One Year (12) $96	You Pay: $89.99 /	Save: $6
Int'l - Two Years (24) $216	You Pay: $170 /	Save: $46 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $99.99 /	Save: $8

☐ **WebLogic Developer's Journal**

U.S. - Two Years (24) Cover: $360	You Pay: $169.99 /	Save: $190 + FREE $198 CD
U.S. - One Year (12) Cover: $180	You Pay: $149 /	Save: $31
Can/Mex - Two Years (24) $360	You Pay: $179.99 /	Save: $180 + FREE $198 CD
Can/Mex - One Year (12) $180	You Pay: $169 /	Save: $11
Int'l - Two Years (24) $360	You Pay: $189.99 /	Save: $170 + FREE $198 CD
Int'l - One Year (12) $180	You Pay: $179 /	Save: $1

☐ **ColdFusion Developer's Journal**

U.S. - Two Years (24) Cover: $216	You Pay: $129 /	Save: $87 + FREE $198 CD
U.S. - One Year (12) Cover: $108	You Pay: $89.99 /	Save: $18
Can/Mex - Two Years (24) $240	You Pay: $159.99 /	Save: $80 + FREE $198 CD
Can/Mex - One Year (12) $120	You Pay: $99.99 /	Save: $20
Int'l - Two Years (24) $264	You Pay: $189 /	Save: $75 + FREE $198 CD
Int'l - One Year (12) $132	You Pay: $129.99 /	Save: $2

☐ **Wireless Business & Technology**

U.S. - Two Years (24) Cover: $144	You Pay: $89 /	Save: $55 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $49.99 /	Save: $22
Can/Mex - Two Years (24) $192	You Pay: $139 /	Save: $53 + FREE $198 CD
Can/Mex - One Year (12) $96	You Pay: $79.99 /	Save: $16
Int'l - Two Years (24) $216	You Pay: $170 /	Save: $46 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $99.99 /	Save: $8

☐ **WebSphere Developer's Journal**

U.S. - Two Years (24) Cover: $360	You Pay: $169.99 /	Save: $190 + FREE $198 CD
U.S. - One Year (12) Cover: $180	You Pay: $149 /	Save: $31
Can/Mex - Two Years (24) $360	You Pay: $179.99 /	Save: $180 + FREE $198 CD
Can/Mex - One Year (12) $180	You Pay: $169 /	Save: $11
Int'l - Two Years (24) $360	You Pay: $189.99 /	Save: $170 + FREE $198 CD
Int'l - One Year (12) $180	You Pay: $179 /	Save: $1

☐ **PowerBuilder Developer's Journal**

U.S. - Two Years (24) Cover: $360	You Pay: $169.99 /	Save: $190 + FREE $198 CD
U.S. - One Year (12) Cover: $180	You Pay: $149 /	Save: $31
Can/Mex - Two Years (24) $360	You Pay: $179.99 /	Save: $180 + FREE $198 CD
Can/Mex - One Year (12) $180	You Pay: $169 /	Save: $11
Int'l - Two Years (24) $360	You Pay: $189.99 /	Save: $170 + FREE $198 CD
Int'l - One Year (12) $180	You Pay: $179 /	Save: $1

*WHILE SUPPILES LAST. OFFER SUBJECT TO CHANGE WITHOUT NOTICE

Subscribe Online Today **www.sys-con.com/2001/sub.cfm**

SYS-CON MEDIA

initial client with .NET, and a future client in a new language altogether.
- ***In SOA, a service is usually "discoverable:"*** This means there is some directory that potential clients can use to discover interface metadata and/or location.

Service-oriented technologies can be applied to many different problems, such as business-to-business data exchange, Web application development, and point-to-point application data exchange. The important architectural requirements for use-of-service technologies are not necessarily the same. Thus, there are two important points to keep in mind:

- We use the term "integration" to refer to the process of making a set of disparate enterprise systems work together to support common purposes. While developing a user-facing application may leverage integration components, enterprise integration and application development are distinct in purpose, scale, and complexity.
- Our interest is specifically in how to apply a service-oriented architecture to the complex problem of sustainable, enterprise-level integration. While we will argue that a benefit of SOI is an enterprise architecture that can be built out on a project basis, our recommendations are about developing the architecture as a whole over time.

With these basic definitions in mind, we can consider the possible benefits of SOI to the enterprise (see Figure 2).

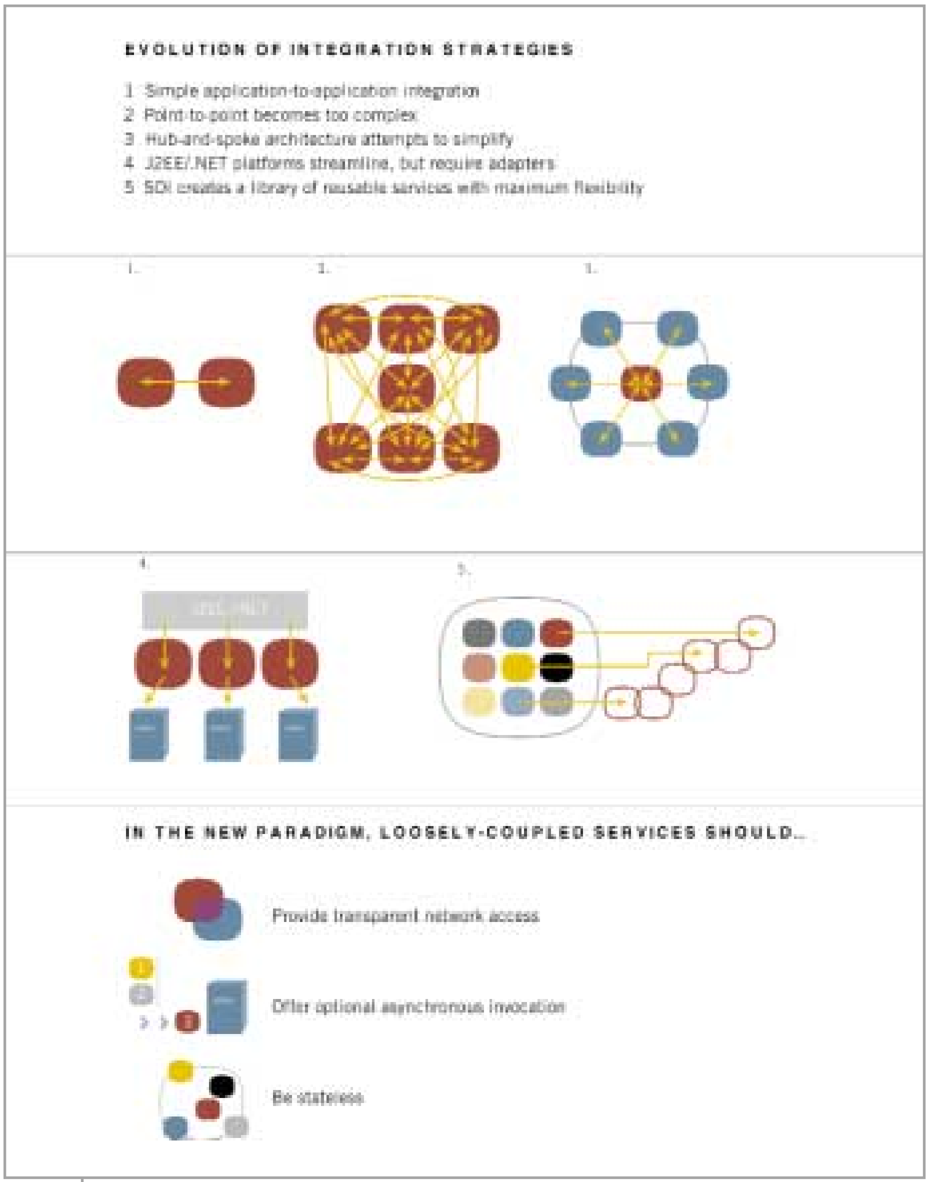

EVOLUTION OF INTEGRATION STRATEGIES
1 Simple application-to-application integration
2 Point-to-point becomes too complex
3 Hub-and-spoke architecture attempts to simplify
4 J2EE/.NET platforms streamline, but require adapters
5 SOI creates a library of reusable services with maximum flexibility
IN THE NEW PARADIGM, LOOSELY-COUPLED SERVICES SHOULD...
Provide transparent network access
Offer optional asynchronous invocation
Be stateless

FIGURE 1 **SOI enables the enterprise to deliver application functionality in short time frames, while supporting future application development**

SOI: Theoretical Benefits

A consistent theme in the section on traditional integration methods is that of cost. The cost can stem from long-term, highly skilled development (custom coding and application servers) or from initial capital expenditure for acquisition and deployment (integration brokers). At the core, SOI holds out the promise of reducing the cost of developing integration architecture:

- As an integration approach, SOI promotes reuse of existing assets with new business applications such as Web self-service, employee portals, and supply-chain management.
- By utilizing a services-based approach, SOI encourages architectures built around services that can be produced on a project-by-project basis and can support reuse over time by multiple clients constructed with a variety of technologies. This leads to earlier return on investment and potentially lower costs over time.
- If built around the standards-based services technologies discussed in succeeding sections, there is a strong possibility that the skill sets and the infrastructures for security, management and the like will cost substantially less than those required by traditional integration approaches.

These benefits are only theoretical, however. Failure to recognize the potential pitfalls of service-oriented architectures can quickly lead to failure on a massive scale. This is where a set of best practices becomes important.

SOI: Best Practices

At least three best practices can be derived directly from our notion of service-oriented architectures:

- ***A repository supports future development:*** A core tenet of SOI is that services can be developed independent of a particular client. In turn, at a future design time, an application or service developer will need to know what services are available, what they do, and how to interface with them. A repository, or directory of service metadata, provides a common place to search and analyze what work has already been done. This is essential to promoting reuse, which is one key component to cost reduction.

- ***A directory service provides flexibility:*** Enterprise integration is a complex undertaking. The underlying hardware and software infrastructures are likely to change over time. Thus, at deployment time affording clients the ability to ask a directory where to find the required service adds a key element of flexibility to the SOI architecture. The directory can be something as simple as a configuration file or environment variable or as sophisticated as an enterprise directory server using UDDI or LDAP.
- ***Client neutrality assists in cost reductions:*** Choose a service technology that is independent of the service's implementation medium. This enables both sets of developers, those writing clients and those creating the actual services, to leverage the technologies with which they are most familiar and/or those technologies most applicable to the job at hand.

"The advent of the information age has brought an increased emphasis on timely access to information"

In the context of SOI specifically, we believe in three additional best practices:

- ***Ensure proper granularity of the service:*** Because services can be developed semi-autonomously, a client developer should not be expected to have intimate knowledge of the underlying enterprise system. Such a requirement slows the project down and risks the client containing implicit dependencies on the service's underlying system. Thus, encapsulation of complexity increases productivity and long-term flexibility to change implementation.

Furthermore, network exchange can be a performance bottleneck due to the overhead of formatting data for the network and reformatting it in the original form on the other side. Thus, a service should take as few trips across the network as possible to accomplish an atomic task. In particular, the fact that it might be easy to wrap a traditional component with a "service interface" does not mean that is the right architectural decision.

- ***Minimize the impact of additional client load:*** You cannot assume knowledge of where or how often a service will be used. Consider carefully whether a service can be productively used in an asynchronous fashion, meaning a client leaves a request on a queue and expects a response later. This architecture allows the service configuration to dictate processing load, not the number of clients. As another example, in many cases an integrated service can be "stateless," meaning only a single exchange of data is necessary to accomplish a task. The service's client application or process has more particular knowledge of its own purpose; that is often the more desirable place to store state information.
- ***Audit service usage:*** Tracking the actual usage of services provides information about which services are still in use, how often, and by whom. In a loosely coupled system this is probably the only realistic mechanism available to understand service interdependencies. In addition, auditing permits legacy services and service versions to be retired, those which receive heavy usage to be optimally deployed, and so forth. Lack of information in these areas will quickly render the enterprise services architecture unmanageable.

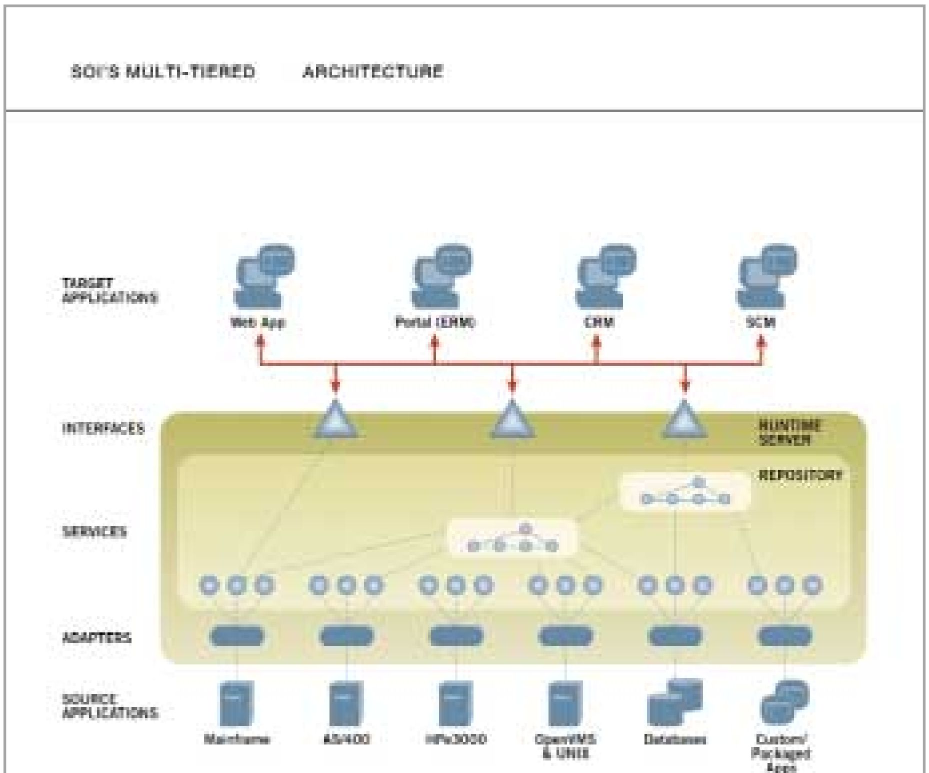


FIGURE 2 **Developing the right service-oriented architecture will enable sustainable, enterprise-level integration**

“Web services are among the most intriguing due to the corresponding development of industry standards”

Service-oriented integration can be pursued with a number of technologies. However, Web services are among the most intriguing due to the corresponding development of industry standards.

SOI and Web Services

Web services standards provide some key building blocks for our new integration paradigm:

- XML is the common, text-based lexical format for data exchange.
- WSDL is an XML-based language for describing service interfaces in a manner that is neutral to client technologies.
- UDDI allows Web services to advertise their existence.
- The related SOAP standard provides standard protocol bindings.

The catch in all of this is that Web services are just a collection of new technology standards. By themselves, these technologies don't constitute a service-oriented architecture, but only enable it. There are still many issues that must be carefully considered in the context of integration:

- ***Reliability and security:*** These features of message-based and some application platform solutions have not been directly addressed. For example, the use of Secure Sockets Layer (SSL) encryption can address basic security requirements, but this does not directly provide message integrity and application-level authentication and authorization capabilities. These issues make the maturation of standards such as WS-Reliability and WS-Security or their equivalents an essential piece of the Web services puzzle.
- ***Transaction support:*** In many cases, this is an important element of integration architecture. Web services transaction support presents an interesting problem because the resources involved could include almost any kind of system that resides almost anywhere. Once again, there are emerging standards, such as WS-Transaction, that should address this shortcoming.
- ***Management:*** Since a service is intended to offer a loosely-coupled solution, this by definition means it is published over a network to any number of clients in any number of environments. Thus, there is no clear-cut method to manage changes to the service over time. The WS-Management and related standards bear watching in this area.

Technology acquisition provides the most promising avenue for addressing these concerns. While careful analysis regarding compatibility with existing enterprise assets will be essential, very little benefit is likely to be derived from a custom implementation of these complex standards.

Applying Web Services to Integration Today

The benefits of the standards-based services paradigm to enterprise development are quite compelling for many departments that need to break the cycle of monolithic application development and find new avenues for extensibility and reuse of existing assets.

The initial architectural consideration is building out services that are client-neutral, abstracted interfaces to existing functionality. This can be applied today in existing projects such that an extensive adoption of SOI can be made more easily in the future. The ability of enterprise applications, integration tools, and adapters to encapsulate the systems they represent should form the basis of one key purchase criterion for each type of product.

As Web services standards for management, security, transactions, and reliability mature, more and more products currently used to support integration efforts will begin to incorporate and interoperate with them. This will allow organizations with existing integration infrastructures to use Web services in a complementary fashion. For those with an existing component-based architecture, the maturation of these standards will permit adoption of the SOI paradigm as a lower-cost option around which to build out the next generation architecture.

In the short term, the best strategy on implementing Web services in support of SOI will be to start with small departmental projects. This will enable your organization to experiment with the SOI methodology presented in this article. In addition, it enables your team to gain experience with the various tools and technologies available today in support of the service-oriented approach to integration. You'll find that the initial projects will be leveraged over and over again as you build out your enterprise-wide set of services in support of current and future business initiatives.

About the Authors

Bill DeForeest is an expert in the development of software that enables customers to implement integration solutions in host-intensive environments. He is responsible for requirement analysis and authoring for WRQ Verastream, as well as strategic and tactical planning. Previously, Bill was a software engineer for LECO Corporation, and was a system administrator with the Spokane Intercollegiate Research and Technology Institute.

■■■ billde@wrq.com

Scott Rosenbloom is a strong proponent of service-oriented integration. He looks for ways to help customers address current integration needs while building a foundation for future initiatives. Scott helps shape the future of the WRQ Verastream product line. Prior to joining WRQ, Scott spent 10 years at IBM in software design, development, and consulting.

■■■ scottr@wrq.com

A new tool for MX professional developers and designers...

Optimizing Web Services Using Java, part I

Generic Java and Web services

What lies behind Web services? Some say the answer depends on the power of the language used in the implementation, in addition to known standards like XML, SOAP, and WSDL. Developing Web services is hard since incorrect use of the language can cause subtle and pernicious errors. What patterns and idioms should we use for simplifying the development process?

WRITTEN BY
JORDAN ANASTASIADE

In this first of two articles, I describe some of the proposed changes to Java and show how they work together to make Java technology a more expressive language for Web services development. In a later article I'll use the Java Web Services Developer Pack (JWSDP 1.3), JAX-RPC 1.1 with its improved schema binding, and the architecture for Basic Profile 1.0, to demonstrate how to design Web services that perform well, how to identify idioms and patterns, and how to optimize Web services performance.

Generic Types

This article describes how generics will improve the design of Web services in Java. So what are generic types? Generics is basically a way to abstract over types. Practically, you can parameterize classes, interfaces, arrays, and methods. Take, for example, a well-known interface, List from java.util package, and try to rewrite it using a generic type T. A simplified version of a generic list interface would look like this:

```
interface List<T> {
  void add(T x);
  Iterator<T> iterator();
}
```

where T is type parameter being a reference to an object of any type from your List. Although the syntax may look similar, generics are not templates as in C++. Let's emphasize the advantages of generics. For instance, if you have a list of String objects you would probably write code like:

```
List firstList = new LinkedList();
firstList.add("something");String s =
(String)firstList.iterator.
    next();
```

What's wrong with this code? Nothing. We've used it all the time as there were no other choices. However, someone working with your list could use it to pull an Integer object out of that list:

```
Integer i = (Integer)
firstList.iterator.next();
```

While the compiler will be happy, due to the cast operation the code will crash at runtime because your list contains only String objects. Wouldn't it be nice if you could tell the compiler what element type your list is? This is exactly what generics allows you to do:

```
List<String> secondList = new
LinkedList<String>();
secondList.add("something");
String s = secondList.iterator.next();
```

The generics code seems to be very similar to the previous code; in fact, you're dealing with the same idiom, but there is a big difference in the declaration of the generic variable. Now, with the generic code, the compiler could check that what you put in or what you pull out of the list is actually the type of object you declared your list to be. The collection List<String> allows only String objects. You could find more about generics and even use, today, a generic-capable compiler from the JSR-14 proposal.

What could we say about the relationship between a Collection of Integer and a Collection of Number where Collection<T> is a parameterized class in a type variable T? Mathematics demonstrates that Collection<Integer> is a subset of <Collection>Number while Integer is a subclass of Number class. *How could we define a subtype relation between different instantiations of the same generic class?*

Let C<T> be a parameterized class where T is a type variable, and A and B two classes, where B is a subclass of A. Is there a subtyping relationship between the two instantiations C<A> and C<B>? While the Programmer class is a subclass of the Employee class, we intuitively accept that List<Programmer > is a subclass of List<Employee >, but we cannot formally define such a relationship. More than that, if you want to declare a variable to hold, alternatively, both instances of List<Programmer > and List<Employee >, you have two options. The first option would be to use the raw type List obtained by the erasure of a parametric type T from the parameterized List<T> class. The second option would be to use a variable of type Object. Both solutions defeat the goals of genericity and the former may also result in unsafe code.

Furthermore, we could consider the Java array as a parameterized class with the type parameter being the type of the array. Between arrays there is already defined a subtyping relationship. For instance, Programmer[] is a subtype of Object[] because Programmer is a subtype of Object. Wouldn't it be nice if a Web service implementation (see Listing 1) never generated a run-time error? Could we define statically safe arrays, arrays that could be checked at compile time?

The solution is based on the notion of *variance*, which introduces a subtype relationship on the set of types defined by a parameterized class C<T>. The variant parameterized type is defined by replacing the type variable T with *wildcards*. Wildcards are useful where no specific knowledge about the type parameter is required. Thus, we can introduce the following definitions.

Covariance

A parameterized class C<T> is said to be covariant in the type parameter T, if for any classes A and B with B a subclass of A, the parameterized type C<B> is a subclass of the parameterized type C<A>. The covariance annotation symbol is a wildcard with an explicit upper bound defined by the syntax

```
? extends T
```

where T is the bound. For example, the type

```
Vector<? extends Number>
```

is said to be covariant in Number and defines the subset of all instantiations of generic class Vector<T> where type variable T is at least of type Number. Covariant means that type variables and parameterized types vary in the same direction (covariance). The parameterized type List<? extends Employee > is a supertype of any parameterized type List<E> where Employee is a supertype of E and could be interpreted as all lists whose elements should be of al least type Employee. For instance, a parameterized type Vector<Integer> could be used when the parameterized type Vector<? extends Number> is needed:

```
 Vector<? extends Number> vn =
new Vector<Integer>();
```

Contravariance:

A parameterized class C<T> is said to be contravariant in the type parameter T, if for any classes A and B with B a subclass of A, the parameterized type C<A> is a subclass of the parameterized type C<B>. The contravariance annotation symbol is a wildcard with an explicit lower bound defined by the syntax:

```
? super T
```

where T is the bound. For instance, Vector<? super A> specializes Vector<? super B> if B specializes A, meaning that the type variables and parameterized types vary in the opposite directions (contra-variance). An Integer is a subtype of Number , whereas Vector<Number> is a subtype of Vector<? super Integer>. Thus, the parameterized type Vector<Number> could be used when the parameterized type Vector<? super Integer> is needed:

```
Vector<? super Integer> vi =
   new Vector<Number>();
```

Bivariance

A parameterized class C<T> is said to be bivariant in the type parameter T, if C<T> is covariant and contravariant in T. The wildcard ? is the bivariance annotation symbol. Vector<?> ranges over all instances of Vector<T> for any T. The wildcard in Vector<?> signifies that this could be a vector of anything. Thus, Vector<? super Integer> is a subtype of Vector<?> and Vector<? extends Integer> is a subtype of Vector<?>.

Invariance

A parameterized class C<T> is said to be invariant in the type parameter T, when C<A> is a subclass of C<B> if and only if A = B. The invariant behavior imposes unnecessary limitations restricting the useful subtyping of generic types. Defining the concept of variance as a subtyping scheme for generic classes, we have introduced a new type of polymorphism called parametric polymorphism. Thus, for any generic class C<T> where T is a type parameter the following subtyping relationships hold true: C<T> is a subtype of C<? super T> which is a subtype of C<?>. The same goes true for covariant type: C<? extends T>.

Generally, variance of parametric types could be defined for any number of parameters. For instance, consider a parameterized class called Pair in type parameters U and V. The Vector covariant in Pair<U,V> could be:

```
Vector<? extends
  Pair<?, ? super Integer>>
```

where the Pair is bivariant in the first element type "?" while the second element type is contravariant in the type parameter Integer.

Could the new variant scheme for generics be applied to any parameterized class? We will see in the next section that some constraints should be defined and applied to variant generic types for the soundness of our Web services design.

Applicability and Usefulness of Generic Types in Web Services Implementation

Consider a simple class hierarchy you define for a Web service implementation describing a company work force. You probably would design classes like employees for sales persons, programmers, managers, etc. (Listing 2). Let's suppose that you have to implement a Web service that calculates salary for a department of the company. You would probably organize the employees of that particular department as a list with a method departmentSalary() of Company class (see Listing 3). Let's suppose further that you want to calculate the salary of sales persons in that particular department using the generic method as defined in Listing 3. You cannot invoke the method with List<Sales> as list of sales persons because your method expects a List<Employee> although Sales is an Employee object. However, with variant generic types you can modify the method signature as follows: List<? extends Employee> meaning that any subclass of Employee class could be used as an actual type parameter. The modified method signature would be:

```
public void departmentSalary
  (List<? extends Employee> employees);
```

There is still a problem. Suppose that your company accepts co-op students during the summer. You could create some temporary employees as a list of co-op students (see Listing 2). Now you can assign to an employees object the object students, because Student is a subclass of Employee class. If the snippet code in Listing 4 were allowed into your Web service implementation, the result would be that your GeneralManager would be stored in a list of students, as a temporary employee, causing headaches for you as a Web service developer. Had we allowed a write operation on a covariant generic type we would have compromised the integrity and consistency of our data structure and the safety and soundness of our Web services. To maintain the static type safety of generic code and to take advantage of the flexibility of variant generic type we have to restrict covariant generic type to read-only operations.

You might think that restricting the type of operation to a read-only one loses the expressiveness of a covariant generic type. For instance the idiom of creating objects using factory methods is well known. Consider a generic interface Factory<T> defined as:

```
public interface Factory<T> {
 T create();
}
```

You might want to define a Web service for the human resources department that manages the company's employees. If you have written classes for the known types of employees:

```
public SalesFactory implements
Factory<Sales> {
 public Sales create() {...}
}
```

you ought to design an easy-to-maintain Web service implementation that could be used for creating any type of employee, even types that are unknown at the time you write your first implementation as in Listing 6. You were able to design a covariant generic Factory<? extends Employee> type that can be used to create any kind of Employee objects. Listing 5 shows that you do not have to modify your implementation, even if your company hires a new employee for a job that did not exist at the time you wrote your Web service.

Suppose that you need a list of employees hired in the last month:

```
public void currentEmployees (List<?
super Employee> list, Employee e);
```

A contravariant generic type like List<? super Employee> is safe, since you put in your list objects of type Employee and the type parameter of List is being defined with ? super Employee as any supertype of Employee type . While the write operations are safe, we have to restrict the contravariant generic type to write-only operations.

Mixed variance parameterizations could be useful as well. Consider, for instance, a generic method that could be used to move the employees from one department to another (see Listing 6). Suppose now that you want to implement a Web service that finds out the best salesperson in your company. Probably you would define a generic Comparator<T> interface. Listing 7 uses a write-only contravariant generic Comparator<? super T> that would be able to compare any two objects. For instance, if you want to sort objects of type T you could define a sort method like:

```
public static <T extends Comparable<?
super T>> void sort(List<T> c);
```

On a bivariant generic class, neither read nor write operations are permitted. You may ask yourself where and how such a type could be useful as long as neither reading nor writing are allowed on such types. For example, based on a list of the departments and a list of all employees per department you want to calculate the total number of workers in your company (see Listing 8). The inner wildcard ? means that the inner List is a bivariant type. You don't use the list for reading out or for writing into, since the list as a bivariant type doesn't allow such operations. You only take the size of the list. The outer List with the type parameter: '? extends List<?>' defines a covariant type in List<?>. You use the read-only operation on the covariant outer List to discover the departments of your company.

Consider another example, with a class Pair<E, R>, where the first parameter defines an employee while the second

WSJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
Altova	www.altova.com		21
Edge East 2004	www.sys-con.com/edge	201-802-3069	56, 57
Ektron	www.ektron.com/ws		9
IBM	ibm.com/websphere/middleware		60
JDJ Store	www.jdjstore.com	888-303-5282	23
LinuxWorld Conference & Expo	www.linuxworldexpo.com		27
LinuxWorld Magazine	www.sys-con.com	201-802-3020	41
Mindreef	www.mindreef.com		5
Mindreef News	www.mindreef.com	603-465-2204	4
MX Developer's Journal	www.sys-con.com/mx/subscription.cfm		47
Parasoft	www.parasoft.com/wsj12	888-305-0041	2
Quest Software	www/java.quest.com/jcsc/ws	800-663-4723	6
SAP	www.sap.com/netweaver	800-880-1727	C3
Strike Iron	www.strikeiron.com		17
SYS-CON Media	www.sys-con.com/suboffer.cfm	888-303-5282	43
SYS-CON Media	www.sys-con.com	888-303-5282	15, 35
Vordel	www.vordel.com/soapbox		13
WebAppCabaret	www.webappcabaret.com/jdj.jsp		31

–Continued from page 7

- Customizable permissions based on user and group permissions
- Easy configuration and delivery to end-user pages
- The ability to share portlets between portal sites
- The ability to scale to large enterprise deployments
- The ability for end users to easily personalize their own portlets and customize where they are used within the portal

The JSR 168 and WSRP industry standards will unquestionably be cornerstones in the foundation of any truly successful portal deployment that aspires to the SOA ideal. However, portal administrators also require a solid infrastructure for creating, customizing, deploying, and reusing portlet application functionality across multiple portal instances. Developers should look for a portal solutions provider with a vision and a long-term track record of developing standards-based portals that encompass all of these building blocks as well.

About the Author

As vice president of product strategy, Edward Anuff is shaping Vignette's go-to-market strategy and maintaining a line of award-winning products and services. He joined Vignette following their acquisition of Epicentric, a portal software company, where he was chairman, cofounder, and chief strategy officer. Since the acquisition, he has taken an active role in the integration of the companies. He is the author of *The Java Sourcebook* (J. Wiley and Sons), one of the first books on the Java programming language.

e-mail: anuff@vignette.com

defines a resource type. If your manager wants to have a list of all employees that need some vacation you can define a method with a signature like:

```
public <E> void employeesNeeds(List<?
extends Pair<? extends E, ?>> e);
```

The List is covariant in Pair<? extends E, ?> type, because it could be using any subtype of Pair<E, R> class. On the other hand Pair class is covariant in the first type parameter E, meaning that the first parameter of Pair<E, R> could be any employee type. The method does not read or write the particular type of resources, hence it is the bivariant annotation symbol "?" for the second parameter of Pair<E, R> class that defines resources.

Conclusion

We've examined the issues involved in supporting variant generic types in Java. A key aim in introducing genericity and variance to the Java programming language is the desire to write general, flexible, and complex Web services where decoupling and reuse are very important goals, while retaining and improving static type safety. Furthermore, variance annotations in class- and interface-type parameters increase the flexibility of subtyping relationships, allowing a better abstraction and maintainability and optimizing Web services as later articles will demonstrate.

Generics increases the readability, maintainability, and safety of our Web services and will be introduced in the next release of the Java programming language (J2SE 1.5 Tiger code name). That release will also include JSR-201 with enumerations, autoboxing for loop enhancements, import of static members, and metadata – features that are easy to use as neither syntax nor semantic restrictions have been imposed on the original language. My next article will demonstrate how to us the JWSDP 1.2, JAX-RPC 1.1 with generics and some of the new features that will make our Web services safer and easier to develop.

References

- Bracha, Gilad; Odersky, Martin; Stoutamire, David; and Wadler, Philip. GJ Specification. Draft document, 1998. http://developer.java.sun.com/developer/earlyAccess/adding_generics/index.html
- Igarashi, Atsushi; and Viroli, Mirko. *On Variance-based Subtyping for Parametric Types.* ECOOP 2002
- Joshua Bloch. (2000). *Effective Java Programming Language Guide.* Java Series, Sun Microsystems. ISBN 0-201-31005-8.

About the Author

Jordan Anastasiade is a professor of computer studies at Seneca College at York University, Toronto campus. He has created industry-leading software products architecting and developing distributed applications. Currently, he is teaching Java technology and Web services.

jordan.anastasiade@senecac.on.ca

Listing 1: Web service implementation – generates runtime error

```
Object[] objectArray = new Integer[1];
objectArray[0] = "zero"; //throws exception at run-time
```

Listing 2: Company class hierarchy implementation

```
public abstract class Employee {
  public abstract int salaryFrom(Company c);
}
public class Sales extends Employee {
  ...
  public int salaryFrom(Company c){ ...}
  ...
}
public class Programmer extends Employee {
  ...
  public int salaryFrom(Company c){ ...}
  ...
}
public class Student extends Employee {
  private List<CoopStudent> coopStudents;
  public int salaryFrom(Company c){ ...}
  ...
}

public class Company {
  int salary = 0;
  public void daySalary(Employee e) {
    salary += e.salaryFrom(this);
  }
}
```

Listing 3: Department salary web service implementation

```
public void
departmentSalary(List<Employee> employees) {

  for (Employee e: employees)
    salary += e.salaryFrom(this);
}
```

Listing 4: Error writing to a covariant type

```
List<Student> students =
                    new LinkedList<Student>();
List<? extends Employee> employees = students;
employees.add(0, new GeneralManager());  // error
```

Listing 5: Using covariant generic type as factory

```
public class HummanResources {

  private Factory<? extends Employee> factory;

  public void
  setFactory(Factory<? extends Employee> f) {
    factory = f;
  }

  /* Generates a new employee object */
  public void newEmployee() {
    Employee e = factory.create();
 ...
  }
}
```

Listing 6: Using covariant and contravariant generic type

```
void move(List<? super Employee> to,
          List<? extends Employee> from);
```

Listing 7: Using contravariant generic type as comparator

```
public interface Comparator<T> {
  int compare(T x, T y);
}

public static <T>
T theBest(List<T> e, Comparator<? super T> c);
```

Listing 8: Using bivariant generic type

```
public  int
all(List<? extends List<?>> departments) {

  int total = 0;
  for (List<?> department : departments) {
    total += department.size();
  }

  return total;
}
```

Download the code at
sys-con.com/webservices

Introducing WS-CAF

More than just transactions

Web services have become the integration platform of choice for enterprise applications. Those applications by the very nature of their enterprise-scale components can be complex in structure, which is compounded by the need to share common data or context across business processes supported by those applications. Those processes may be very long lived, and may contain periods of inactivity, for example, where constituent services require user interactions.

WRITTEN BY
JIM WEBBER &

MARK LITTLE

In response to these issues, WS-CAF (Web Services Composite Application Framework) was publicly released in July 2003 after almost two years of effort, and has broad industry support from companies such as Iona, Oracle, Sun, and a host of others, and is now under the care of an OASIS standardization effort through the WS-CAF Technical Committee. The WS-CAF specifications are a suite of protocols designed to provide the necessary framework for composing Web services into larger aggregate business processes. Given that WS-CAF is the first framework of its kind to make its way into standardization, it's important to understand the principles underpinning it.

This article provides a high-level view of WS-CAF starting from the bottom up, explaining the layered architecture of the trio of specifications that comprise WS-CAF, and demonstrating how each of the specifications can be used in its own right or as a whole to provide a rich framework for building reliable composite applications.

WS-Context (WS-CTX)

The ability to scope units of work (known as activities) is a requirement of a variety of aspects of distributed applications. In order to correlate the work of multiple Web services within the same activity, it's necessary to propagate additional information – the context – to each participating service. The *context* contains information such as a unique ID that allows a series of operations to share a common outcome, and is propagated in a SOAP header block whenever application messages are transmitted between component services. The reliable management of the contexts that provide distributed application scope is addressed by the WS-Context specification.

The purpose of a context is to allow multiple individual Web services to enter a relationship by sharing certain common attributes as an externally modeled entity. Typical reasons for Web services to share context include common security domains where multiple Web services execute within the scope of a single authorized session, or common outcome negotiation where each party within the activity needs to know whether each of the other participants successfully completed his or her work.

The structure of a context is application specific (as we shall see, WS-CoordinationFramework and WS-Transaction-Management both extend the basic WS-Context context for their own purposes), but contains at a minimum a unique ID in the form of a URI. Web services are identified as participants in the activity by including the context in an application message's SOAP header block (see Listing 1).

In general terms, a context defines basic information about the activity structure. It contains information necessary for multiple Web services to be associated with the same activity, which may be dynamically updated by services as the application or process makes progress. Activities are managed by the *context service*, which maintains a repository of shared contexts associated with execution environments. Whenever messages are exchanged within the scope of an activity, the context service can supply the associated context, which may then be propagated with those messages. The Context Service also manages hierarchies of contexts to support nesting and concurrency.

As we have seen, the core context propagation framework provides a generic context structure that enables an activity to be uniquely identified so that work can be correlated. Additionally, it supports application- and service-specific extensions to the context, structure. To facilitate this, the context consists of:

- ***A mandatory URI identifier called <context-identifier>:*** Guarantees global uniqueness for an individual activity
- ***An optional list of child activities:*** The <child-contexts> element
- ***A <timeout> element:*** Indicates how long the context information is valid

In addition to the context service, each Web service participating in an activity may register an Activity Lifecycle Service (or ALS) with the Context Service, which allows that service to be informed about the lifetime of the context. As we shall see, the ALS is the key component in utilizing WS-Context as the base protocol for other higher-level protocols. During execution, when a context is required for the activity associated with the current execution environment, the Context Service calls each registered ALS and obtains additional content for the basic context from it; from this it eventually assembles the entire context document that can be propagated.

The relationship between ALS and context service, application services, and applications is shown in Figure 1.

WS-Context does not mandate how contexts are actually created, but the canonical route is via the Activity Lifecycle services, which "plug-in" to the Context Service. In this respect, the exact structure and semantics of an activity are defined by the combination of ALSs that have been associated with the activity. For example, a Context Service may have a transaction ALS and security ALS registered with it, so that when an activity is started, any context that is created will contain any necessary transaction and security information.

WS-Coordination Framework (WS-CF)

WS-CF is the middle layer in the WS-CAF set of specifications and provides an extensible framework that supports a wide range of different coordination protocols (e.g., two-phase or three-phase commit).

While WS-Coordination Framework is ostensibly similar to WS-Coordination, the main differentiator is that WS-CF defines more of the coordinator's architecture than WS-Coordination (which leaves most things up to the services that use it). For example, in WS-CF you can actually inquire as to the status of a coordinator without having to know the details of the protocol (and its implementation). In many ways, WS-CF can be considered a superset of the WS-Coordination.

Figure 2 illustrates how individual Web services as well as composite applications can register as participants with a coordinator, which takes over responsibility for context management and notifying participants of the outcome of a series of related Web services executions. As the figure shows, a coordinator can register itself with another coordinator and become a participant, thereby improving interoperability.

WS-Transaction Management (WS-TXM)

WS-TXM builds on WS-CF to provide transactional coordination. Figure 3 illustrates the layering of WS-TXM protocols. WS-TXM defines a set of pluggable transaction protocols that can be used with the coordinator to negotiate a set of actions for all participants to execute based on the outcome of a series of related Web services executions. The executions are related through the use of shared context (scopes) that can be nested (parent-child relationships) and concurrent.

WS-TXM actually embodies three separate extended transaction protocols. Like WS-Transaction and BTP, WS-TXM provides models that are designed to accommodate multiple use cases, from tightly-coupled intranet-based transactions (TX-ACID), to Internet-scale, long-lived transactions (TX-LRA), to business process–oriented transactions (TX-BP).

ACID Transactions

This model is designed to support interoperability of existing transaction processing systems via Web services, since such systems already form the backbone of enterprise class applications. Although ACID transactions may not be suitable for all Web services, they are most definitely suitable for some, and particularly high-value interactions such as those involved in finance. As a result, the ACID transaction model defined in WS-TXM has been designed with interoperability in mind. In the ACID model, each activity is bound to the scope of a transaction, so that the end of an activity automatically triggers the termination (commit or rollback) of the associated transaction.

Long Running Activities (LRA)

The LRA protocol is designed for those business interactions that are long in duration. Within this model, all work performed within the scope of an application should be compensatable. Therefore, an application's work is either performed successfully or undone. How individual Web services perform their work and ensure it can be undone if compensation is required is an implementation choice. The LRA model simply defines the triggers for compensation actions and the conditions under which those triggers are executed.

There is a caveat to this model though. Application services may not be compensatable (e.g., an application-level service that prints and mails checks), or the ability to compensate may be transient. The LRA model allows applications to combine services that can be compensated with those that cannot be compensated. Obviously, by mixing the two service types the user may end up with a business activity that will ultimately not be undone by the LRA model, but which may require outside (application-specific) compensation.

The LRA model defines a protocol actor called a *compensator* that operates on behalf of a service to undo the work it performs within the scope of an LRA. How

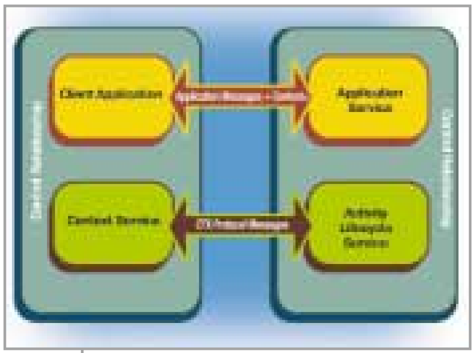


FIGURE 1 | Relationship between ALS, context service, applications, and application services

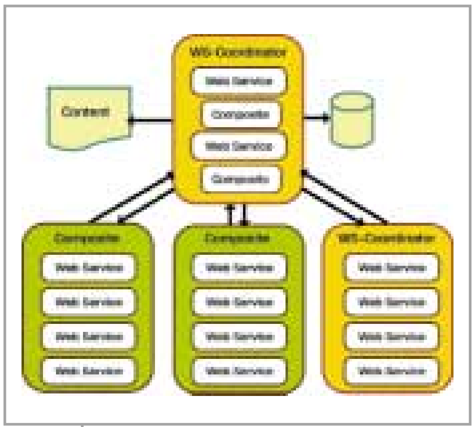


FIGURE 2 | Relationship of coordinator to Web services and composite applications

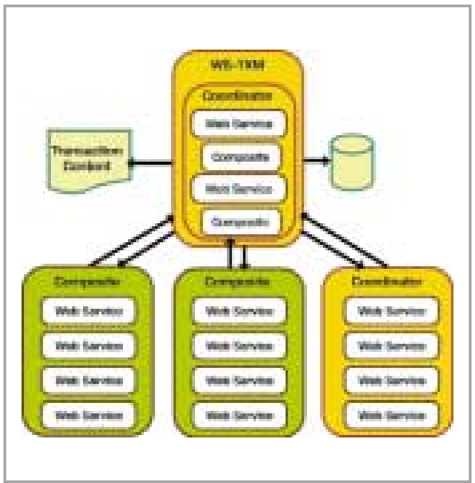


FIGURE 3 | Relationship of transactions to coordination framework compensation is carried out will obviously be dependent upon the service; compensation work may be carried out by other LRAs which themselves have compensators.

When a service performs work that may later have to be compensated within the scope of an LRA, it enlists a compensator participant with the LRA coordinator. The coordinator will send the compensator one of the following messages when the activity terminates:

- ***Success:*** The activity has completed successfully. If the activity is nested, then compensators may propagate that outcome to the enclosing LRA.
- ***Fail:*** The activity has not completed. All compensators that are registered with the LRA will be invoked to perform compensation in reverse order. The coordinator forgets about all compensators that indicated they operated correctly. Otherwise, compensation may be attempted again or a compensation violation has occurred and must be logged.

LRAs may be used both sequentially and concurrently, where the termination of an LRA signals the start of some other unit of work within an application. However, LRAs are units of compensatable work and an application may have as many units of work operating simultaneously as it needs to accomplish its tasks. Furthermore, the outcome of work within LRAs may determine how other LRAs are terminated.

An application can be structured so that LRAs are used to assemble units of compensatable work and then held in the active state while the application performs other work in the scope of different (concurrent or sequential) LRAs. Only when the right subset of work (LRAs) is arrived at by the application will that subset be confirmed; all other LRAs will be told to cancel (complete in a failure state).

Business Process (BP)

The BP protocol is significantly different from any of the other transaction models we have seen to date (and there is no directly comparable model in either WS-Transaction or BTP). This model is specifically aimed at tying heterogeneous transaction domains together into a single business-to-business transaction. For example, with the BP model it's possible to have a long-running business transaction span messaging, workflow, and traditional ACID transactions, allowing enterprises to leverage their existing IT investment.

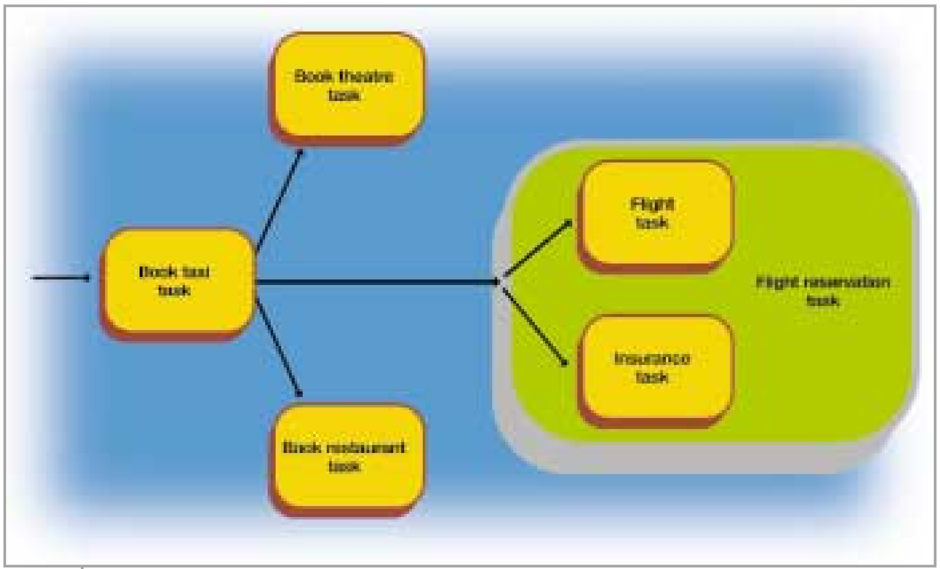


FIGURE 4 | **Business processes and tasks for a travel agent**

In the business process transaction model, all parties involved in a business process reside within business domains, which may themselves use business processes to perform work. Business process transactions are responsible for managing interactions between these domains. A business process is split into business tasks and each task executes within a specific business domain. A business domain may itself be subdivided into other business domains recursively.

Each domain may represent a different transaction model if such a federation of models is more appropriate to the activity. Each business task (which may be modelled as a scope) may provide implementation-specific countereffects in the event the enclosing scope must cancel. Furthermore, the controlling application may periodically request that all business domains checkpoint their state so that they can either be consistently rolled back to that checkpoint by the application or restarted from the checkpoint in the event of a failure.

Figure 4 shows an online travel agent interacting with its suppliers, each of which resides in its own business domain. The work necessary to obtain each component is modelled as a separate task. In this example, the Flight Reservation task is actually composed of two subtasks – one gets the flight and the other gets the necessary travel insurance.

In this example, the user may interact synchronously with the travel agent to build up the required details of the holiday. Or, the user may submit an order (possibly with a list of alternate requirements, such as destinations, dates, etc.) to the agent, who will call back when it has been filled. Likewise, the travel agent then submits orders to each supplier, requiring them to call back when each component is available (or is known to be unavailable).

Business domains are instructed to perform work within the scope of a global business process. The business process has an overall manager that may be informed by individual tasks when they have completed their work or it may periodically communicate with each task to determine its current status. In addition, each task may make checkpoints of its progress so if a failure occurs, it may be restarted from that point rather than having to start from the beginning. A business process can either terminate in a confirmed (successful) manner, in which case all of the work requested will have been performed, or it will terminate in a cancelled (unsuccessful) manner, in which case all of the work will be undone.

If it cannot be undone, then this fact must be logged.

Summary

From a distance, WS-CAF may be misinter-

Comparison Between OASIS BTP and WS-Coordination/Transaction

WS-CAF is not the only transactional coordination protocol for Web services. Indeed, in the past we've seen OASIS BTP and IBM/Microsoft/BEA WS-Coordination and WS-Transaction. To help illustrate the features of WS-CAF, it is instructive to take a look at the factorization and features of the prior efforts.

OASIS BTP was the first transaction protocol to gain real traction for Web services. It consists of a single API that supports two distinct transaction models, known as atom and cohesion. The atom model is a straightforward two-phase protocol where all participants in a transaction see the same outcome, although BTP does not impose any semantics on what action a particular participant takes on receipt of an outcome message (an atom may or may not be ACID). The cohesion model is more complex, and allows the set of participants to change throughout the duration of the transaction, up until the point when the confirmation protocol executes. However, unlike the atom model, BTP cohesions may deliver different outcome messages to individual participants, based on the combination of responses from participants and some business logic.

Similarly, WS-Transaction has two transaction models: atomic transactions require ACID semantics and mandate that resources are locked for the transaction's duration. Business activities, on the other hand, are designed for use in long-running transactions. They ensure that any updates to state in a system are made immediately, significantly reducing the period during which locks must be held. WS-Transaction has no notion of a two-phase commit for a business activity because commits are made immediately on receipt of the associated messages. If a failure occurs, a business activity runs compensating actions to restore data to a consistent form.

Underpinning WS-Transaction is WS-Coordination, which provides a generic mechanism for context creation and coordination and is extended through protocol plug-ins that provide domain-specific coordination facilities.

Figure 5 highlights the two key differences between the specifications. The most striking feature is that each offers different transaction models at the uppermost layers, but it is important to note that the WS-Coordination layer in the WS-Transaction/WS-Coordination stack is also available for applications to build on. In the WS-CAF stack, the WS-Context layer is also exposed for use.

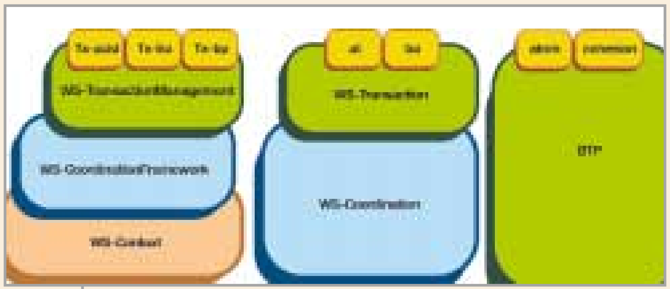


FIGURE 5 | **A side-by-side comparison of WS-CAF, WS-Coordination and WS-Transaction, and OASIS BTP**

preted simply as the industry's third attempt at designing a transaction management solution for Web services. However, while one aspect of WS-CAF does address the kind of extended transaction models that are crucial for Web services reliability, there is actually much more to WS-CAF than just transactions. WS-CAF also provides generic context-management and service-coordination frameworks that can form the basis of composite applications, processes, and workflows. These features are exposed to Web services–based applications and can be tailored to build protocols that are specific to particular applications domains.

References

- Webber J., and Little M.C. (May 2003) "Introducing WS-Coordination, part 1" ***Web Services Journal***, Vol. 3, Issue 5.
- Little M.C. and Webber J. (June 2003) "Introducing WS-Transaction, part 2," *Web Services Journal*. Vol. 3, Issues 6–7.
- Dalal, S., et al. (January 2003). "Coordinating Business Transactions on the Web." IEEE Internet Computing Special Edition on Web Services.

About the Authors

Dr. Mark Little is the transactions architect for Arjuna Technologies Limited, a spin-off from Hewlett-Packard that develops reliable middleware for J2EE and Web services. Prior to this, Mark was an architect for HP Middleware where he led the transactions teams. He is one of the primary authors for the OMG Activity Service Specification; a member of the expert group for the work in J2EE: JSR 95, JSR 117; and is the specification lead for JSR 156 (Java API for XML Transactions).
■■■ mark.little@arjuna.com

Dr. Jim Webber is an architect and Web services fanatic at Arjuna Technologies, where he works on Web services transactioning and Grid computing technology. Prior to joining Arjuna Technologies, he was the lead developer with Hewlett-Packard working on their BTP-based Web Services Transactions product – the industry's first Web services transaction solution.
■■■ jim.webber@arjuna.com

Listing 1: Sample WS-Context context embedded in a SOAP Header block

```
<env:Envelope xmlns:env="http://www.w3.org/2002/12/soap-envelope">
 <env:Header>
  <wsctx:context

xmlns:wsctx="http://www.webservicestransactions.org/schemas/wsas/2003/03"
    timeout="0">
     <wsctx:context-identifier>
       http://www.arjuna.com/ws-ctx/123:abc:456:def
     </wsctx:context-identifier>
     <!-- Other protocols can extend context here -->
   </wsctx:context>
 </env:Header>
 <env:Body>
   <m:Message xmlns:m="http://example.org/MessageSchema"
    <m:…
   </m:Message>
 </env:Body>
</env:Envelope>
```

Download the code at sys-con.com/webservices

4TH Annual International Deve

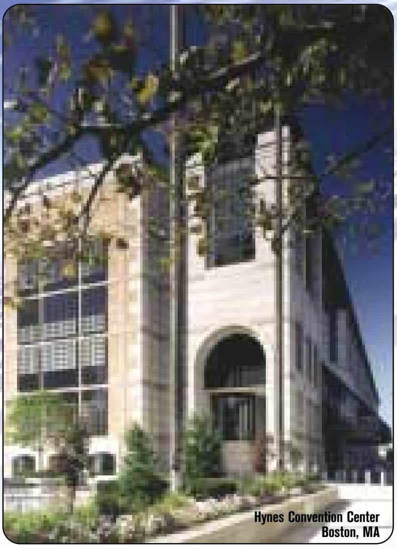
edge 2004 EAST

DEVELOPMENT TECHNOLOGIES EXCHANGE

February 24-26, 2004

Hynes Convention Center, Boston, MA

ARCHITECTING JAVA, .NET, WEB SERVICES, MX, XML, AND OPEN SOURCE



Hynes Convention Center
Boston, MA

PROVIDING DEVELOPERS THE LATEST IN:

- Application Integration
- Desktop Java
- Mobility
- ASP.NET
- VS.NET
- JavaServer Faces
- SOA
- Interoperability
- Rich Internet Applications

Preregister for ***FREE*** full-day tutorials featuring the hottest Java, .NET, XML, Web Services, and Security issues. (ADVANCED REGISTRATION REQUIRED)

FREE Exhibition Floor (ADVANCED REGISTRATION REQUIRED)
Meet with industry-leading companies showcasing the newest products and services.

- Product Demonstrations Theater
- Opening Night Reception
- Internet Café
- Case Study Forum

Register By December 19, 2003 **SAVE** Up To **$400**

MX ® WebSphere XML .net

For more information
visit www.sys-con.com/edge
call 201 802-3069
e-mail events@sys-con.com

Over 100 participating companies will display and demonstrate over 300 developer products and solutions.

Over 3,000 C-Level Executives, Systems Integrators, System Architects, Developers, and Project Managers will attend the conference and expo.

Over 100 sessions – training, certifications, seminars, case studies, and panel discussions – promise to deliver real-world benefits, the industry pulse, and proven strategies.

Full-Day Tutorials Targeting

- Java
- .NET
- XML
- Web Services
- Open Source
- Security

Conference program available online!
www.sys-con.com/edge

loper Conference & Expo

JOIN THE EXCHANGE

Become a part of the Development Technologies Exchange in Boston this February. Learn from the leading voices in the i-Technology world, see the latest developments from the companies that push the envelope of innovation, and network with your peers.

The Edge 2004 East Conference Program holds the information you need to excel in today's competitive environment.

.NET Track

- .NET Compact Framework Performance Tips and Tricks
- Best Practices and Techniques for Building Secure ASP.NET Applications
- Using the Enterprise Instrumentation Framework
- .NET Framework: Exploring What's New in the Base Class Library for "Whidbey"
- Microsoft Office 2003: A Solutions Platform
- BizTalk Server 2004 Technical Drilldown
- Moving your Architecture to .NET
- Introduction to VisualBasic
- Intro to C#

Web Services Track

- Enterprise Integration & Web Services
- WS-BPEL
- UDDI
- Web Services Progress Report
- Building Interoperable WS using WS-I Basic Profile
- Web Services Progress Report
- Identity Management
- Middleware Integration

XML Track

- XML Schema Applications
- XML & Web Services Security
- XPath 2.0 and XSLT 2.0
- XML Schema Languages
- XML & the Enterprise
- XForms
- Universal Business Language
- XQuery
- Using XML for EAI - Best Practices

Java Track

- Enterprise Architecture & Open Source Java Generics Primer
- J2EE v1.4:
- JXTA and J2ME-Enabled Wireless Mobile Devices Apache Axis ANT/JUnit
- Simplifying J2EE Applications
- SWT
- The Evolution of J2EE
- JDK 1.5 Empowering Java and RSS for Blogging FREE One Day Microsoft .NET Tutorial

Attend a **FREE** One Day Tutorial Presented By Microsoft®

.NET–The Smart Client Perspective

For the business world, one of the most exciting promises of the information age is the opportunity to provide employees with access to information and the tools to act on that information whenever and wherever they need to. To a certain extent, that promise has already been achieved. Today, most businesspeople work on PCs that provide access to information, applications, and resources far beyond the boundaries of their local machine.

There are limits, however. Today's Internet model for information and application distribution assumes access to a network connection, but ubiquitous Web connectivity still lies in the future. And some computing tasks require robust functionality that can only be provided efficiently by "rich" client applications that reside on the local computer.

A challenge arises when your organization requires both the flexibility and immediacy that comes with online access to data and applications, and the full functionality of traditional client software.

The answer: smart client software.

This day long tutorial focuses on developing and deploying smart client applications.

Seating is limited. Full Conference Attendees will receive Priority Seating for all Tutorials, all other seating is on a first come basis.

VISIT www.sys-con.com/edge FOR DETAILS and REGISTER TODAY

edge 2004 EAST DEVELOPMENT TECHNOLOGIES EXCHANGE

The program, including topics, and times are subject to change. Please refer to www.sys-con.com for all updates.

Reactivity and Mindreef Partner to Police Rogue Web Services

(Belmont, CA and Boston, MA) – Reactivity, a leader in delivering instant and sustainable XML Web services security solutions, and Mindreef, Inc., a leading provider of Web services diagnostics technology, have partnered to offer a free, co-branded version of Mindreef's SOAPscope Lite, a focused functionality version of their Web Services Diagnostics System. Available from the Reactivity Web site, the SOAPscope "Lite" tool enables application architects, security analysts, and IT operations staff to easily detect rogue Web services on a network and get their Web services applications up and running more quickly by capturing and revealing the content of XML and SOAP messages.

The Reactivity XML Firewall provides extensive audit and message logs and integrated SNMP and SMTP alerts that enable developers, security architects, and network operators to instantly identify and rapidly resolve XML Web services security and interoperability issues. www.mindreef.com, www.reactivity.com

Solstice Software Delivers Integration Testing Suite

(Orlando, FL) – Solstice Software, Inc., a pioneer in behind-the-screens message testing, has announced Integra Enterprise 4.0, a commercialized, enterprise-class integration testing suite. By enabling visibility and control of the messaging backbone of mainframe integration, Web services, and other complex software integration projects, Integra Enterprise solves the most vexing problems that face integration project managers, senior managers, and QA staff.

Solstice Software provides a suite of automated testing and simulation tools designed to streamline and improve reliability of integration projects. Integra Enterprise's comprehensive protocol library and unique focus on messaging give project teams the ability to go "behind-the-screens" and unearth problems buried in messages and files that can bring down critical systems. www.solsticesoftware.com

Intalio Extends BPMS with Page Designer

(San Mateo, CA) – Intalio, Inc., a business process management company, has announced Intalio|n^3 2.5, a new version of their business process management system.

The Intalio|n^3 Page Designer, a WYSIWYG editor for the development of rich, Web-based end-user interfaces, provides 100% code coverage, integration with Systinet's Web services platform, integration of Corticon's business rule engine, and a new, customizable user interface to enhance the productivity of business analysts. www.intalio.com

Wysdom and Cape Clear Software Deliver Web Services–Based Mobile Integration

(Toronto) – Wysdom Inc., a provider of mobile network operating system software, has announced a working partnership with Cape Clear Software to provide mobile operators with an easier means of integrating and delivering third-party content and services.

The combination of Wysdom's MAP-OS mobile network operating system software and Cape Clear's Web Services technology will simplify integration of telecom applications, enabling the rapid launch of new business services and applications with built-in support for the major telecom standards such as Parlay X.

Built around Web services, Cape Clear 4.5 enables organizations to create business services, the building blocks of business integration. By combining Cape Clear with Wysdom's MAP-OS mobile network operating system software, operators can integrate systems and rapidly deliver highly secure data applications for SMS and MMS services. www.wysdom.com, www.capeclear.com

Radiant Logic Launches Identity Infrastructure Platform

(Novato, CA) – Radiant Logic, Inc., a provider of virtual directory solutions, has launched RadiantOne 3.0, a flexible identity infrastructure designed to accelerate the deployment of identity management (IdM) initiatives.

RadiantOne 3.0 is a comprehensive identity infrastructure that combines virtual directory technology with directory storage, LDAP routing and load balancing, synchronization services, and tools for identity infrastructure planning, design, and simulation. www.radiantlogic.com

Fiorano Software Rebrands Its Business Integration Infrastructure Stack

(Los Gatos, CA) – Fiorano Software Inc., a provider of enterprise-class integration solutions, has rebranded its business integration product line, "Tifosi," as the "Fiorano Business Integration Suite."

The Fiorano Business Integration Suite will encompass Fiorano's complete integration Infrastructure Stack of products under one brand name. Components of the stack include Fiorano ESB, FioranoMQ, Fiorano Business Service Composer, Fiorano BPM, Fiorano monitoring and management tools, and Fiorano adapters. www.fiorano.com

Unifact Corporation Delivers Business Intelligence Software Platform

(Boston) – Unifact Corporation has released UniViz Analytics, a browser-based ad hoc data analysis tool targeted at both business analysts and casual business users.

UniViz Analytics is a software solution designed to leverage Microsoft .NET's rich-client Web technology. Supporting emerging Web services standards for analytics, UniViz Analytics delivers a blend of connectivity, interactivity, and collaboration. The software provides an easy-to-use, Web-delivered application with the power of OLAP and ad hoc analysis tools, and the dashboard-style interface preferred by casual users. www.unifact.com

Composite Software Announces First EII Product

(San Mateo, CA) – Composite Software today has released the Composite Information Server, an Enterprise Information Integration (EII) product that allows corporate users unfettered access to the information they need, regardless of the complexities caused by a variety of formats, locations and systems. The product creates an information infrastructure layer that allows corporate IT to deliver custom views of information from disparate sources for use by executives, employees, customers and partners. By presenting all enterprise data as if it exists in one location and in any familiar format, the Composite Information Server allows enterprises to maximize their return on information. www.compositesw.com

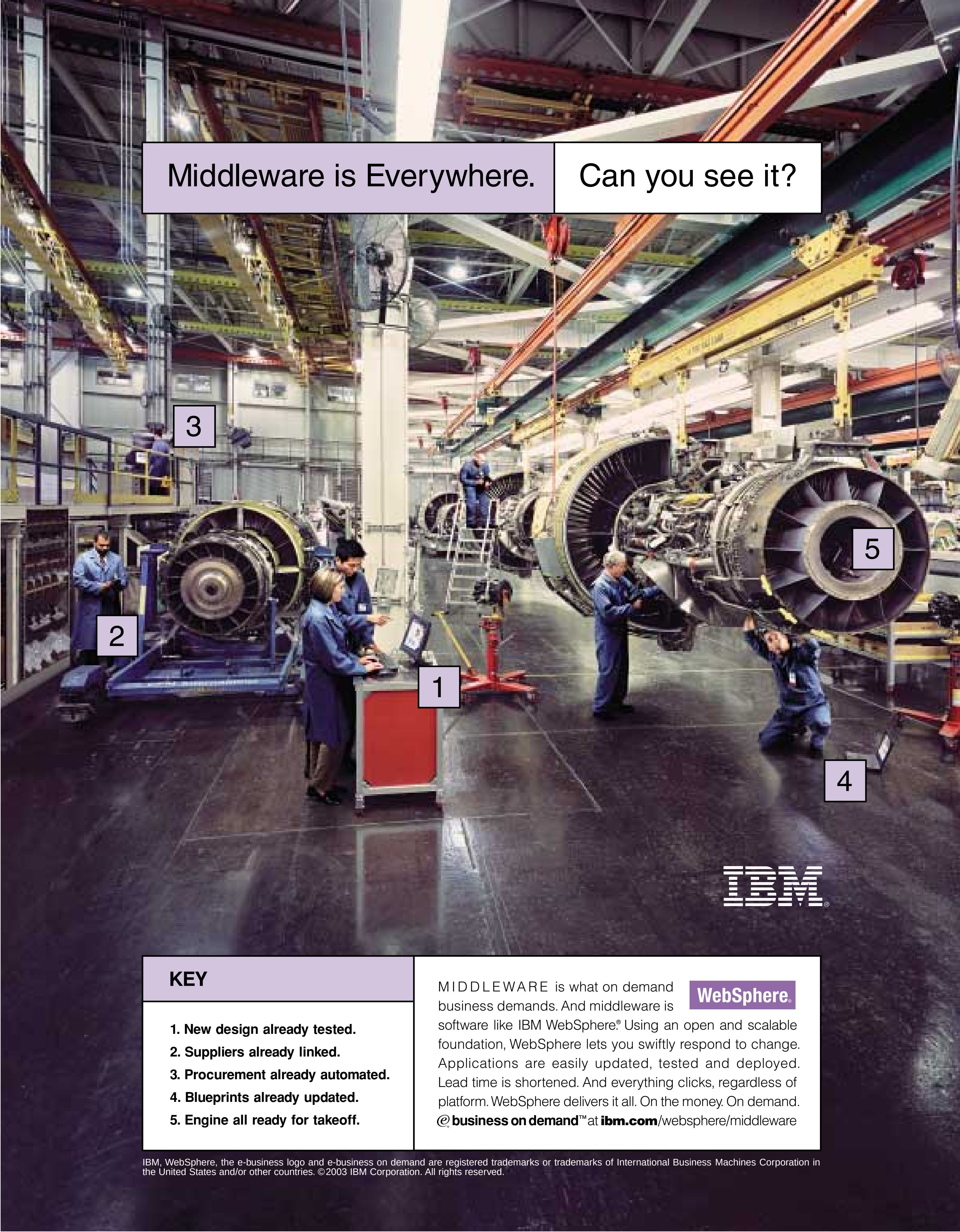
FINALLY, BUSINESS SOLUTIONS THAT WORK WITH EXISTING TECHNOLOGIES AND NONEXISTENT BUDGETS.

You need to get more out of what you have. We have just the thing: solutions based on our open technology platform, SAP NetWeaver™. Because it's preconfigured to work with your current IT investments – and it's fully operable with .NET, J2EE and IBM WebSphere – SAP NetWeaver reduces the need for custom integration. That lowers your total cost of ownership for your entire IT landscape and gets you quicker ROI. Everything a CIO wants (and a CFO didn't think was possible). Visit sap.com/netweaver or call 800 880 1727 for details.

THE BEST-RUN BUSINESSES RUN SAP

SAP®

Middleware is Everywhere. Can you see it?



IBM

KEY

1. **New design already tested.**
2. **Suppliers already linked.**
3. **Procurement already automated.**
4. **Blueprints already updated.**
5. **Engine all ready for takeoff.**

WebSphere

MIDDLEWARE is what on demand business demands. And middleware is software like IBM WebSphere® Using an open and scalable foundation, WebSphere lets you swiftly respond to change. Applications are easily updated, tested and deployed. Lead time is shortened. And everything clicks, regardless of platform. WebSphere delivers it all. On the money. On demand.

e **business on demand**™ at **ibm.com**/websphere/middleware